

Informix® Guide to SQL

Reference

Informix Dynamic Server, Version 7.3
Informix Dynamic Server with Advanced Decision Support and Extended Parallel Options, Version 8.2
Informix Dynamic Server, Developer Edition, Version 7.3
Informix Dynamic Server, Workgroup Edition, Version 7.3

February 1998
Part No. 000-4365

Published by INFORMIX® Press

Informix Software, Inc.
4100 Bohannon Drive
Menlo Park, CA 94025-1032

Copyright © 1981-1998 by Informix Software, Inc. or its subsidiaries, provided that portions may be copyrighted by third parties, as set forth in documentation. All rights reserved.

The following are worldwide trademarks of Informix Software, Inc., or its subsidiaries, registered in the United States of America as indicated by “®,” and in numerous other countries worldwide:

Answers OnLine™; INFORMIX®; Informix®; Illustra™; C-ISAM®; DataBlade®; Dynamic Server™; Gateway™; NewEra™

All other names or marks may be registered trademarks or trademarks of their respective owners.

Documentation Team: Smita Joshi, Geeta Karmarkar, Jennifer Leland, Barbara Nomiya

RESTRICTED RIGHTS/SPECIAL LICENSE RIGHTS

Software and documentation acquired with US Government funds are provided with rights as follows: (1) if for civilian agency use, with Restricted Rights as defined in FAR 52.227-19; (2) if for Dept. of Defense use, with rights as restricted by vendor's standard license, unless superseded by negotiated vendor license as prescribed in DFAR 227.7202. Any whole or partial reproduction of software or documentation marked with this legend must reproduce the legend.

Table of Contents

Introduction

About This Manual 3

 Types of Users 3

 Software Dependencies 4

 Assumptions About Your Locale. 4

 Demonstration Databases 4

New Features 5

 New Features in Version 7.3 5

 New Features in Version 8.2 6

Documentation Conventions 7

 Typographical Conventions 7

 Icon Conventions 8

 Command-Line Conventions 10

 Sample-Code Conventions. 13

Additional Documentation 14

 On-Line Manuals 14

 Printed Manuals 14

 Error Message Files 14

 Documentation Notes, Release Notes, Machine Notes 15

 Related Reading 16

Compliance with Industry Standards 17

Informix Welcomes Your Comments 18

Chapter 1 System Catalog

Objects That the System Catalog Tables Track 1-3

Using the System Catalog 1-4

 Accessing the System Catalog. 1-9

 Updating System Catalog Data 1-9

Structure of the System Catalog	1-10
SYSBLOBS	1-12
SYSCHECKS	1-13
SYSCOLAUTH	1-14
SYSCOLDEPEND	1-15
SYSCOLUMNS	1-15
SYSCONSTRAINTS	1-18
SYSDEFAULTS	1-19
SYSDEPEND	1-21
SYSDISTRIB	1-22
SYSEXTCOLS	1-23
SYSEXTDFILES	1-24
SYSEXTERNAL	1-24
SYSFRAGAUTH	1-25
SYSFRAGMENTS	1-27
SYSINDEXES	1-29
SYSNEWDEPEND	1-32
SYSOBJSTATE	1-32
SYSOPCLSTR	1-33
SYSPROCAUTH	1-36
SYSPROCBODY	1-36
SYSPROCEDURES	1-38
SYSPROCPLAN	1-39
SYSREFERENCES	1-40
SYSREPOSITORY	1-40
SYSROLEAUTH	1-41
SYSSYNONYMS	1-42
SYSSYNTABLE	1-42
SYSTABAUTH	1-43
SYSTABLES	1-44
SYSTRIGBODY	1-47
SYSTRIGGERS	1-48
SYSUSERS	1-49
SYSVIEWS	1-50
SYSVIOLATIONS	1-50
System Catalog Map	1-51
Information Schema	1-55
Generating the Information Schema Views	1-55
Accessing the Information Schema Views	1-56
Structure of the Information Schema Views.	1-56

Chapter 2**Data Types**

Database Data Types	2-3
Summary of Data Types	2-3
BYTE	2-5
CHAR(<i>n</i>)	2-6
CHARACTER(<i>n</i>)	2-8
CHARACTER VARYING(<i>m,r</i>)	2-8
DATE	2-8
DATETIME	2-9
DEC	2-13
DECIMAL	2-13
DOUBLE PRECISION	2-15
FLOAT(<i>n</i>)	2-15
INT	2-15
INTEGER	2-16
INTERVAL	2-16
MONEY(<i>p,s</i>)	2-19
NCHAR(<i>n</i>)	2-20
NUMERIC(<i>p,s</i>)	2-21
NVARCHAR(<i>m,r</i>)	2-21
REAL	2-21
SERIAL(<i>n</i>)	2-21
SMALLFLOAT	2-22
SMALLINT	2-23
TEXT	2-23
VARCHAR(<i>m,r</i>)	2-25
Data Type Conversions	2-27
Converting from Number to Number	2-27
Converting Between Number and CHAR	2-28
Converting Between DATE and DATETIME	2-29
Range of Operations on DATE, DATETIME, and INTERVAL	2-29
Manipulating DATETIME Values	2-31
Manipulating DATETIME with INTERVAL Values	2-32
Manipulating DATE with DATETIME and INTERVAL Values.	2-33
Manipulating INTERVAL Values.	2-34
Multiplying or Dividing INTERVAL Values	2-35

Chapter 3**Environment Variables**

Types of Environment Variables	3-5
Where to Set Environment Variables in UNIX	3-6
Environment Variables at the System Prompt	3-6
Environment Variables in an Environment-Configuration File	3-6
Environment Variables at Login Time.	3-6
Manipulating Environment Variables in UNIX	3-7
Setting Environment Variables in an Environment-Configuration File	3-7
Setting Environment Variables at Login Time in UNIX	3-8
Syntax for Setting Environment Variables in UNIX	3-8
Unsetting Environment Variables in UNIX	3-9
Modifying the Setting of an Environment Variable in UNIX	3-9
Viewing Your Environment Variable Settings in UNIX	3-10
Checking Environment Variables in UNIX with the chkenv Utility	3-11
Rules of Precedence in UNIX	3-12
Where to Set Environment Variables in Windows NT	3-12
Manipulating Environment Variables in Windows NT	3-13
Setting Environment Variables for Native Windows Applications	3-13
Setting Environment Variables for Command-Prompt Utilities in Windows NT	3-14
Rules of Precedence for Environment Variables in Windows NT	3-17
List of Environment Variables	3-18
Environment Variables	3-22
ARC_DEFAULT	3-22
ARC_KEYPAD	3-23
CPFIRST.	3-24
DBANSIWARN	3-25
DBBLOBBUF	3-26
DBCENTURY	3-26
DBDATE	3-29
DBDELIMITER	3-32
DBEDIT	3-32
DBFLTMASK	3-33
DBLANG	3-33
DBMONEY.	3-35
DBONPLOAD.	3-36
DBPATH.	3-37
DBPRINT	3-39
DBREMOTECMD	3-40
DBSPACETEMP	3-41
DBTIME.	3-43

DBUPSPACE	3-46
DELIMIDENT	3-46
ENVIGNORE	3-47
FET_BUF_SIZE	3-48
IFX_DIRECTIVES	3-48
INFORMIXC.	3-50
INFORMIXCONRETRY	3-50
INFORMIXCONTIME	3-51
INFORMIXDIR	3-52
INFORMIXKEYTAB	3-53
INFORMIXOPCACHE	3-54
INFORMIXSERVER	3-54
INFORMIXSHMBASE	3-55
INFORMIXSQLHOSTS	3-56
INFORMIXSTACKSIZE	3-57
INFORMIXTERM	3-58
INF_ROLE_SEP.	3-59
NODEFDAC.	3-59
ONCONFIG	3-60
OPTCOMPIND.	3-61
PATH	3-62
PDQPRIORITY	3-62
PLCONFIG	3-64
PSORT_DBTEMP	3-64
PSORT_NPROCS	3-65
SQLEXEC.	3-67
SQLRM	3-68
SQLRMDIR	3-68
TERM	3-69
TERMCAP	3-69
TERMINFO	3-70
THREADLIB.	3-71
Index of Environment Variables	3-71

Appendix A The stores7 Database

Glossary

Index

Introduction

About This Manual.	3
Types of Users	3
Software Dependencies	4
Assumptions About Your Locale.	4
Demonstration Databases	4
New Features.	5
New Features in Version 7.3	5
New Features in Version 8.2	6
Documentation Conventions	7
Typographical Conventions	7
Icon Conventions	8
Comment Icons	8
Feature, Product, and Platform Icons	9
Compliance Icons	10
Command-Line Conventions	10
How to Read a Command-Line Diagram	12
Sample-Code Conventions	13
Additional Documentation	14
On-Line Manuals	14
Printed Manuals	14
Error Message Files	14
Documentation Notes, Release Notes, Machine Notes	15
Related Reading	16
Compliance with Industry Standards	17
Informix Welcomes Your Comments.	18

R

ead this introduction for an overview of the information provided in this manual and for an understanding of the documentation conventions used.

About This Manual

This manual includes information regarding individual system catalog tables, data types, and environment variables used by Informix products. It also includes information on designing and using ANSI-compliant databases and a description of the demonstration database, **stores7**.

Types of Users

This manual is for the following users:

- Database users
- Database administrators
- Database server administrators
- Database-application programmers

This manual assumes that you have the following background:

- A working knowledge of your computer, your operating system, and the utilities that your operating system provides
- Some experience working with relational databases or exposure to database concepts

If you have limited experience with relational databases, SQL, or your operating system, refer to the *Getting Started* manual for your database server for a list of supplementary titles.

Software Dependencies

This manual assumes that you are using one of the following database servers:

- Informix Dynamic Server, Version 7.3
- Informix Dynamic Server, Developer Edition, Version 7.3
- Informix Dynamic Server, Workgroup Edition, Version 7.3
- Dynamic Server with AD and XP Options, Version 8.2

Assumptions About Your Locale

Informix products can support many languages, cultures, and code sets. All culture-specific information is brought together in a single environment, called a GLS (Global Language Support) locale.

This manual assumes that you are using the default locale, **en_us.8859-1**. This locale supports U.S. English format conventions for dates, times, and currency. In addition, this locale supports the ISO 8859-1 code set, which includes the ASCII code set plus many 8-bit characters such as é, è, and ñ.

If you plan to use nondefault characters in your data or your SQL identifiers, or if you want to conform to the nondefault collation rules of character data, you need to specify the appropriate nondefault locale.

For instructions on how to specify a nondefault locale, additional syntax, and other considerations related to GLS locales, see the [Informix Guide to GLS Functionality](#).

Demonstration Databases

The DB-Access utility, which is provided with your Informix database server products, includes a demonstration database called **stores7** that contains information about a fictitious wholesale sporting-goods distributor. You can use SQL scripts provided with DB-Access to derive a second database, called **sales_demo**. This database illustrates a dimensional schema for data-warehousing applications. Sample command files are also included for creating and populating these databases.

Many examples in Informix manuals are based on the **stores7** demonstration database. The **stores7** database is described in detail and its contents are listed in this manual.

The scripts that you use to install the demonstration databases reside in the **\$INFORMIXDIR/bin** directory on UNIX platforms and the **%INFORMIXDIR%\bin** directory on Windows NT platforms. For a complete explanation of how to create and populate the **stores7** demonstration database, refer to the [DB-Access User Manual](#). For an explanation of how to create and populate the **sales_demo** database, refer to the [Informix Guide to Database Design and Implementation](#).

New Features

The following sections describe new database server features relevant to this manual. For a comprehensive list of new features, see the release notes for your database server.

New Features in Version 7.3

Most of the new features for Version 7.3 of Informix Dynamic Server fall into five major areas:

- Reliability, availability, and serviceability
- Performance
- NT-specific features
- Application migration
- Manageability

Several additional features affect connectivity, replication, and the optical subsystem.

This manual includes information about the following new features:

- Greater network security: the INFORMIXKEYTAB environment variable specifies the location of the **keytab** file used by the DCE-GSS communication-support module at the time a client connects to an Informix database server.
- Performance Improvements: the IFX_DIRECTIVES environment variable determines whether the optimizer allows query optimization from within a query.

New Features in Version 8.2

This manual describes the following new features that have been implemented in Version 8.2 of Dynamic Server with AD and XP Options:

- Support for storing CREATE INDEX statements and other information for general-key indexes.
- Support for tracking external tables through new system catalog tables.
- Support for identifying whether or not table fragments are distributed through a hashing scheme as well as the names of the columns that are hashed.
- Support for recording whether or not an index is a unique index or a non-unique index.

This manual also discusses the following features, which were introduced in Version 8.1 of Dynamic Server with AD and XP Options:

- Fragmentation of table data across multiple computers
- The CASE expression in certain Structured Query Language (SQL) statements
- Nonlogging tables
- External tables for high-performance loading and unloading
- Coserver groups (cogroups) for centralized administration of coservers
- Dbslices for centralized administration of storage spaces

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other Informix manuals.

The following conventions are covered:

- Typographical conventions
- Icon conventions
- Command-line conventions
- Sample-code conventions

Typographical Conventions

This manual uses the following standard set of conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

Convention	Meaning
KEYWORD	All keywords appear in uppercase letters in a serif font.
<i>italics</i>	Within text, new terms and emphasized words appear in italics. Within syntax diagrams, values that you are to specify appear in italics.
boldface	Identifiers (names of classes, objects, constants, events, functions, program variables, forms, labels, and reports), environment variables, database names, filenames, table names, column names, icons, menu items, command names, and other similar terms appear in boldface.
<code>monospace</code>	Information that the product displays and information that you enter appear in a monospace typeface.

(1 of 2)



Convention	Meaning
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
◆	This symbol indicates the end of feature-, product-, platform-, or compliance-specific information within a table or section.
→	This symbol indicates a menu item. For example, “Choose Tools→Options ” means choose the Options item from the Tools menu.

(2 of 2)

***Tip:** When you are instructed to “enter” characters or to “execute” a command, immediately press RETURN after you type the indicated information on your keyboard. When you are instructed to “type” the text or to “press” other keys, you do not need to press RETURN.*

Icon Conventions


Throughout the documentation, you will find text that is identified by several different types of icons. This section describes these icons.

Comment Icons

Comment icons identify warnings, important notes, or tips. This information is always displayed in italics.








Icon	Description
	The <i>warning</i> icon identifies vital instructions, cautions, or critical information.
	The <i>important</i> icon identifies significant information about the feature or operation that is being described.

(1 of 2)

Icon	Description
	The <i>tip</i> icon identifies additional details or shortcuts for the functionality that is being described.
(2 of 2)	

Feature, Product, and Platform Icons




Feature, product, and platform icons identify paragraphs that contain feature-specific, product-specific, or platform-specific information.

Icon	Description
	Identifies information that is specific to Informix Dynamic Server with Advanced Decision Support and Extended Parallel Options.
	Identifies information that is specific to the INFORMIX-ESQL/C product.
	Identifies information that relates to the Informix Global Language Support (GLS) feature.
	Identifies information that is specific to Dynamic Server and its editions. However, in some cases, the identified section applies only to Informix Dynamic Server and not to Informix Dynamic Server, Workgroup and Developer Editions. Such information is clearly identified.
	Identifies information that is specific to UNIX platforms.
	Identifies information that is specific to Informix Dynamic Server, Workgroup and Developer Editions.
	Identifies information that is specific to the Windows NT environment.

These icons can apply to a row in a table, one or more paragraphs, or an entire section. If an icon appears next to a section heading, the information that applies to the indicated feature, product, or platform ends at the next heading at the same or higher level. A ♦ symbol indicates the end of the feature-, product-, or platform-specific information that appears within a table or a set of paragraphs within a section.

Compliance Icons

Compliance icons indicate paragraphs that provide guidelines for complying with a standard.

Icon	Description
	Identifies information that is specific to an ANSI-compliant database.
	Identifies information that is an Informix extension to ANSI SQL-92 entry-level standard SQL.
	Identifies functionality that conforms to X/Open.

These icons can apply to a row in a table, one or more paragraphs, or an entire section. If an icon appears next to a section heading, the compliance information ends at the next heading at the same or higher level. A ♦ symbol indicates the end of compliance information that appears in a table row or a set of paragraphs within a section.

Command-Line Conventions




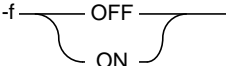
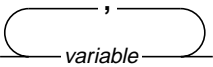
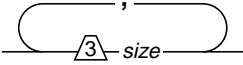
This section defines and illustrates the format of commands that are available in Informix products. These commands have their own conventions, which might include alternative forms of a command, required and optional parts of the command, and so forth.

Each diagram displays the sequences of required and optional elements that are valid in a command. A diagram begins at the upper-left corner with a command. It ends at the upper-right corner with a vertical line. Between these points, you can trace any path that does not stop or back up. Each path describes a valid form of the command. You must supply a value for words that are in italics.

You might encounter one or more of the following elements on a command-line path.

Element	Description
command	This required element is usually the product name or other short word that invokes the product or calls the compiler or preprocessor script for a compiled Informix product. It might appear alone or precede one or more options. You must spell a command exactly as shown and use lowercase letters.
<i>variable</i>	A word in italics represents a value that you must supply, such as a database, file, or program name. A table following the diagram explains the value.
-flag	A flag is usually an abbreviation for a function, menu, or option name or for a compiler or preprocessor argument. You must enter a flag exactly as shown, including the preceding hyphen.
.ext	A filename extension, such as .sql or .cob , might follow a variable that represents a filename. Type this extension exactly as shown, immediately after the name of the file. The extension might be optional in certain products.
(. , ; + * - /)	Punctuation and mathematical notations are literal symbols that you must enter exactly as shown.
' '	Single quotes are literal symbols that you must enter as shown.
<div>Privileges p. 5-17</div> <div>Privileges</div>	A reference in a box represents a subdiagram. Imagine that the subdiagram is spliced into the main diagram at this point. When a page number is not specified, the subdiagram appears on the same page.

(1 of 2)

Element	Description
	A shaded option is the default action.
	Syntax within a pair of arrows indicates a subdiagram.
	The vertical line terminates the command.
	A branch below the main path indicates an optional path. (Any term on the main path is required, unless a branch can circumvent it.)
	A loop indicates a path that you can repeat. Punctuation along the top of the loop indicates the separator symbol for list items.
	A gate ($\sqrt{3}$) on a path indicates that you can only use that path the indicated number of times, even if it is part of a larger loop. Here you can specify <i>size</i> no more than three times within this statement segment.

(2 of 2)

How to Read a Command-Line Diagram

Figure 1 shows a command-line diagram that uses some of the elements that are listed in the previous table.

Figure 1
Example of a Command-Line Diagram



To construct a command correctly, start at the top left with the command. Then follow the diagram to the right, including the elements that you want. The elements in the diagram are case sensitive.

Figure 1 diagrams the following steps:

1. Type the word `setenv`.
2. Type the word `INFORMIXC`.
3. Supply either a compiler name or pathname.
After you choose *compiler* or *pathname*, you come to the terminator.
Your command is complete.
4. Press RETURN to execute the command.

Sample-Code Conventions

Examples of SQL code occur throughout this manual. Except where noted, the code is not specific to any single Informix application development tool. If only SQL statements are listed in the example, they are not delimited by semicolons. For instance, you might see the code in the following example:

```
CONNECT TO stores7
...

DELETE FROM customer
      WHERE customer_num = 121
...

COMMIT WORK
DISCONNECT CURRENT
```

To use this SQL code for a specific product, you must apply the syntax rules for that product. For example, if you are using the Query-language option of DB-Access, you must delimit multiple statements with semicolons. If you are using an SQL API, you must use EXEC SQL at the start of each statement and a semicolon (or other appropriate delimiter) at the end of the statement.



Tip: *Ellipsis points in a code example indicate that more code would be added in a full application, but it is not necessary to show it to describe the concept being discussed.*

For detailed directions on using SQL statements for a particular application development tool or SQL API, see the manual for your product.

Additional Documentation

For additional information, you might want to refer to the following types of documentation:

- On-line manuals
- Printed manuals
- Error message files
- Documentation notes, release notes, and machine notes
- Related reading

On-Line Manuals

An Answers OnLine CD that contains Informix manuals in electronic format is provided with your Informix products. You can install the documentation or access it directly from the CD. For information about how to install, read, and print on-line manuals, see the installation insert that accompanies Answers OnLine.

Printed Manuals

To order printed manuals, call 1-800-331-1763 or send email to moreinfo@informix.com. Please provide the following information when you place your order:

- The documentation that you need
- The quantity that you need
- Your name, address, and telephone number

Error Message Files

Informix software products provide ASCII files that contain all of the Informix error messages and their corrective actions. For a detailed description of these error messages, refer to *Informix Error Messages* in Answers OnLine.

UNIX

To read the error messages on UNIX, use the following commands.

Command	Description
finderr	Displays error messages on line
rofferr	Formats error messages for printing

**WIN NT**

To read error messages and corrective actions on Windows NT, use the **Informix Find Error** utility. To display this utility, choose **Start→Programs→Informix** from the Task Bar. ◆

Documentation Notes, Release Notes, Machine Notes

In addition to printed documentation, the following sections describe the on-line files that supplement the information in this manual. Please examine these files before you begin using your database server. They contain vital information about application and performance issues.

UNIX

On UNIX platforms, the following on-line files appear in the **SINFORMIXDIR/release/en_us/0333** directory.

On-Line File	Purpose
SQLRDOC_x.y	The documentation-notes file for your version of this manual describes features that are not covered in the manual or that have been modified since publication.
SERVERS_x.y	The release-notes file describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.
IDS_x.y	The machine-notes file describes any special actions that are required to configure and use Informix products on your computer. Machine notes are named for the product described.



WIN NT

The following items appear in the Informix folder. To display this folder, choose **Start→Programs→Informix** from the Task Bar.

Item	Description
Documentation Notes	This item includes additions or corrections to manuals, along with information about features that may not be covered in the manuals or that have been modified since publication.
Release Notes	This item describes feature differences from earlier versions of Informix products and how these differences might affect current products. This file also contains information about any known problems and their workarounds.

Machine notes do not apply to Windows NT platforms. ♦

Related Reading

The following publications provide additional information about the topics that are discussed in this manual. For a list of publications that provide an introduction to database servers and operating-system platforms, refer to your *Getting Started* manual.

- *A Guide to the SQL Standard* by C. J. Date with H. Darwen (Addison-Wesley Publishing, 1993)
- *Understanding the New SQL: A Complete Guide* by J. Melton and A. Simon (Morgan Kaufmann Publishers, 1993)
- *Using SQL* by J. Groff and P. Weinberg (Osborne McGraw-Hill, 1990)

UNIX

This manual assumes that you are familiar with your computer operating system. If you have limited UNIX system experience, consult your operating-system manual or a good introductory text before you read this manual. The following texts provide a good introduction to UNIX systems:

- *Introducing the UNIX System* by H. McGilton and R. Morgan (McGraw-Hill Book Company, 1983)
- *Learning the UNIX Operating System* by G. Todino, J. Strang, and J. Peek (O'Reilly & Associates, 1993)

WIN NT

- *A Practical Guide to the UNIX System* by M. Sobell (Benjamin/Cummings Publishing, 1989)
- *UNIX System V: A Practical Guide* by M. Sobell (Benjamin/Cummings Publishing, 1995) ♦

If you are using Windows NT and have limited Windows NT system experience, consult your operating-system manual or a good introductory text before you read this manual. The following texts provide an introduction to Windows NT:

- *Using Windows NT Workstation 3.51* by Paul Sanna (Que, 1996)
- *Microsoft Windows NT Resource Kit* by Russ Blake (Microsoft Press, 1995)
- *NT Server Management and Control* by Kenneth L. Spencer (Prentice-Hall, 1995)
- *Windows NT Administration* by Marshall Brain and Shay Woodard (Prentice-Hall, 1994)
- *Windows NT Network Programming* by Ralph Davis (Addison-Wesley, 1994)
- *Inside Windows NT* by Helen Custer (Microsoft Press, 1993) ♦

Compliance with Industry Standards

The American National Standards Institute (ANSI) has established a set of industry standards for SQL. Informix SQL-based products are fully compliant with SQL-92 Entry Level (published as ANSI X3.135-1992), which is identical to ISO 9075:1992. In addition, many features of Informix database servers comply with the SQL-92 Intermediate and Full Level and X/Open SQL CAE (common applications environment) standards.

Informix Welcomes Your Comments

Please tell us what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about corrections or clarifications that you would find useful. Include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

Write to us at the following address:

Informix Software, Inc.
SCT Technical Publications Department
4100 Bohannon Drive
Menlo Park, CA 94025

If you prefer to send email, our address is:

`doc@informix.com`

Or send a facsimile to the Informix Technical Publications Department at:

650-926-6571

We appreciate your feedback.

System Catalog

Objects That the System Catalog Tables Track	1-3
Using the System Catalog	1-4
Accessing the System Catalog	1-9
Updating System Catalog Data	1-9
Structure of the System Catalog	1-10
SYSBLOBS	1-12
SYSCHECKS	1-13
SYSCOLAUTH	1-14
SYSCOLDEPEND	1-15
SYSCOLUMNS	1-15
SYSCONSTRAINTS	1-18
SYSDEFAULTS	1-19
SYSDEPEND	1-21
SYSDISTRIB	1-22
SYSEXTCOLS	1-23
SYSEXTDFILES	1-24
SYSEXTERNAL	1-24
SYSFRAGAUTH	1-25
SYSFRAGMENTS	1-27
SYSINDEXES	1-29
SYSNEWDEPEND	1-32
SYSOBJSTATE	1-32
SYSOPCLSTR	1-33
SYSPROCAUTH	1-36
SYSPROCBODY	1-36
SYSPROCEDURES	1-38
SYSPROCPLAN	1-39
SYSREFERENCES	1-40
SYSREPOSITORY	1-40

SYSROLEAUTH	1-41
SYSSYNONYMS	1-42
SYSSYNTABLE	1-42
SYSTABAUTH	1-43
SYSTABLES	1-44
SYSTRIGBODY	1-47
SYSTRIGGERS	1-48
SYSUSERS	1-49
SYSVIEWS	1-50
SYSVIOLATIONS	1-50
System Catalog Map	1-51
Information Schema	1-55
Generating the Information Schema Views	1-55
Accessing the Information Schema Views	1-56
Structure of the Information Schema Views	1-56
The tables Information Schema View	1-57
The columns Information Schema View	1-57
The sql_languages Information Schema View	1-59
The server_info Information Schema View	1-59

The system catalog consists of tables that describe the structure of the database. Each system catalog table contains specific information about an element in the database.

This chapter details the following material:

- Using the system catalog
- Structure of the system catalog
- System catalog map
- Information Schema

Objects That the System Catalog Tables Track

The system catalog tables track the following objects:

- Tables and constraints
- Views
- Triggers
- Authorized users and privileges
- Stored procedures

The system catalog tables are generated automatically when you create a database, and you can query them as you would query any other table in the database. For a newly created database, the system catalog tables for the database reside in a common area of the disk called a dbspace. All tables in the system catalog have the prefix **sys** (for example, the **systables** system catalog table).

Using the System Catalog

The database server accesses the system catalog constantly. Each time an SQL statement is processed, the database server accesses the system catalog to determine system privileges, add or verify table names or column names, and so on. For example, the following CREATE SCHEMA block adds the **customer** table, with its respective indexes and privileges, to the **stores7** database. This block also adds a view, **california**, that restricts the *view* in the **customer** table to only the first and last names of the customer, the company name, and the phone number for all customers who reside in California.

```
CREATE SCHEMA AUTHORIZATION maryl
CREATE TABLE customer
    (customer_num SERIAL(101), fname CHAR(15), lname CHAR(15), company CHAR(20),
    address1 CHAR(20), address2 CHAR(20), city CHAR(15), state CHAR(2),
    zipcode CHAR(5), phone CHAR(18))
GRANT ALTER, ALL ON customer TO cathl WITH GRANT OPTION AS maryl
GRANT SELECT ON CUSTOMER TO public
GRANT UPDATE (fname, lname, phone) ON customer TO howe
CREATE VIEW california AS
    SELECT fname, lname, company, phone FROM customer WHERE state = 'CA'
CREATE UNIQUE INDEX c_num_ix ON customer (customer_num)
CREATE INDEX state_ix ON customer (state);
```

To process this CREATE SCHEMA block, the database server first accesses the system catalog to verify the following information:

- The new table and view names do not already exist in the database. (If the database is ANSI compliant, the database server verifies that the table and view names do not already exist for the specified owners.)
- The user has permission to create the tables and grant user privileges.
- The column names in the CREATE VIEW and CREATE INDEX statements exist in the **customer** table.

In addition to verifying this information and creating two new tables, the database server adds new rows to the following system catalog tables:

- **systables**
- **syscolumns**
- **sysviews**

- **systabauth**
- **syscolauth**
- **sysindexes**

The following two new rows of information are added to the **systables** system catalog table after the CREATE SCHEMA block is run, as [page 1-4](#) shows.

tabname	customer
owner	maryl
partnum	16778361
tabid	101
rowsize	134
ncols	10
nindexes	2
nrows	0
created	01/26/1998
version	1
tabtype	T
locklevel	P
npused	0
fextsize	16
nextsize	16
flags	0
site	
dbname	
tabname	california
owner	maryl
partnum	0
tabid	102
rowsize	134
ncols	4
nindexes	0
nrows	0
created	01/26/1998
version	0
tabtype	V
locklevel	B
npused	0
fextsize	0
nextsize	0
flags	0
site	
dbname	

Each table recorded in the **sysables** system catalog table is assigned a **tabid**, a system-assigned sequential ID number that uniquely identifies each table in the database. The system catalog tables receive **tabid** numbers 1 through 24, and the user-created tables receive **tabid** numbers that begin with 100.

The CREATE SCHEMA block adds 14 rows to the **syscolumns** system catalog table. These rows correspond to the columns in the table **customer** and the view **california**, as the following example shows.

colname	tabid	colno	coltype	collength	colmin	colmax
customer_num	101	1	262	4		
fname	101	2	0	15		
lname	101	3	0	15		
company	101	4	0	20		
address1	101	5	0	20		
address2	101	6	0	20		
city	101	7	0	15		
state	101	8	0	2		
zipcode	101	9	0	5		
phone	101	10	0	18		
fname	102	1	0	15		
lname	102	2	0	15		
company	102	3	0	20		
phone	102	4	0	18		

In the **syscolumns** system catalog table, each column within a table is assigned a sequential column number, **colno**, that uniquely identifies the column within its table. In the **colno** column, the **fname** column of the **customer** table is assigned the value 2 and the **fname** column of the view **california** is assigned the value 1. The **colmin** and **colmax** columns contain no entries. These two columns contain values when a column is the first key in a composite index or is the only key in the index, has no null or duplicate values, and the UPDATE STATISTICS statement has been run.

The rows that the following example shows are added to the **sysviews** system catalog table. These rows correspond to the CREATE VIEW portion of the CREATE SCHEMA block.

```
tabidseqviewtext
102 0  create view 'maryl'.california (customer_num, fname, lname, company
102 1  ,address1,address2,city,state,zipcode,phone) as select x0.custom
102 2  er_num, x0.fname, x0.lname, x0.company, x0.address1, x0.address2
102 3  ,x0.city, x0.state, x0.zipcode, x0.phone from 'maryl'.customer
102 4  x0 where (x0.state = 'CA');
```

The **sysviews** system catalog table contains the CREATE VIEW statement that creates the view. Each line of the CREATE VIEW statement in the current schema is stored in this table. In the **viewtext** column, the **x0** that precedes the column names in the statement (for example, **x0.fname**) operates as an alias name that distinguishes among the same columns that are used in a self-join.

The CREATE SCHEMA block also adds rows to the **systabauth** system catalog table. These rows correspond to the user privileges granted on **customer** and **california** tables, as the following example shows.

grantor	grantee	tabid	tabauth
maryl	public	101	su-idx--
maryl	cathl	101	SU-IDXAR
maryl	nhowe	101	--*-----
	maryl	102	SU-ID---

The **tabauth** column of this table specifies the table-level privileges granted to users on the **customer** and **california** tables. This column uses an 8-byte pattern, such as **s** (select), **u** (update), ***** (column-level privilege), **i** (insert), **d** (delete), **x** (index), **a** (alter), and **r** (references), to identify the type of privilege. In this example, the user **nhowe** has column-level privileges on the **customer** table.

If the **tabauth** privilege code is uppercase (for example, **S** for select), the user who is granted this privilege can also grant it to others. If the **tabauth** privilege code is lowercase (for example, **s** for select), the user who has this privilege cannot grant it to others.

In addition, three rows are added to the **syscolauth** system catalog table. These rows correspond to the user privileges that are granted on specific columns in the **customer** table, as the following example shows.

grantor	grantee	tabid	colno	colauth
maryl	nhowe	101	2	-u-
maryl	nhowe	101	3	-u-
maryl	nhowe	101	10	-u-

The **colauth** column specifies the column-level privileges that are granted on the **customer** table. This column uses a 3-byte pattern, such as **s** (select), **u** (update), and **r** (references), to identify the type of privilege. For example, the user **nhowe** has update privileges on the second column (because the **colno** value is 2) of the **customer** table (indicated by **tabid** value of 101).

The CREATE SCHEMA block adds two rows to the **sysindexes** system catalog table. These rows correspond to the indexes created on the **customer** table, as the following example shows.

idxname	c_num_ix	state_ix
owner	maryl	maryl
tabid	101	101
idxtype	U	D
clustered		
part1	1	8
part2	0	0
part3	0	0
part4	0	0
part5	0	0
part6	0	0
part7	0	0
part8	0	0
part9	0	0
part10	0	0
part11	0	0
part12	0	0
part13	0	0
part14	0	0
part15	0	0
part16	0	0
levels		
leaves		
nunique		
clust		

In this table, the **idxtype** column identifies whether the created index is unique or a duplicate. For example, the index **c_num_ix** that is placed on the **customer_num** column of the **customer** table is unique.

Accessing the System Catalog

Normal user access to the system catalog is read only. Users with Connect or Resource privileges cannot alter the system catalog. They can, however, access data in the system catalog tables on a read-only basis using standard SELECT statements. For example, the following SELECT statement displays all the table names and corresponding table ID numbers of user-created tables in the database:

```
SELECT tabname, tabid FROM systables WHERE tabid > 99
```



Warning: Although user **informix** and DBAs can modify most system catalog tables (only user **informix** can modify **systables**), Informix strongly recommends that you do not update, delete, or insert any rows in them. Modifying the system catalog tables can destroy the integrity of the database. Informix supports the use of the ALTER TABLE statement to modify the size of the next extent of system catalog tables.

Updating System Catalog Data

The optimizer in Informix database servers determines the most efficient strategy for executing SQL queries. The optimizer allows you to query the database without having to fully consider which tables to search first in a join or which indexes to use. The optimizer uses information from the system catalog to determine the best query strategy.

If you use the UPDATE STATISTICS statement to update the system catalog, you can ensure that the information provided to the optimizer is current. When you delete or modify a table, the database server does not automatically update the related statistical data in the system catalog. For example, if you delete rows in a table with the DELETE statement, the **nrows** column in the **systables** system catalog table, which holds the number of rows for that table, is not updated. The UPDATE STATISTICS statement causes the database server to recalculate data in the **systables**, **sysdistrib**, **syscolumns**, and **sysindexes** system catalog tables. After you run UPDATE STATISTICS, the **systables** system catalog table holds the correct value in the **nrows** column. If you use the medium or high mode with the UPDATE STATISTICS statement, the **sysdistrib** system catalog table holds the updated data-distribution data after you run UPDATE STATISTICS.

Whenever you modify a table extensively, use the UPDATE STATISTICS statement to update data in the system catalog. For more information on the UPDATE STATISTICS statement, see the [Informix Guide to SQL: Syntax](#).

Structure of the System Catalog

The following system catalog tables describe the structure of the Informix database.

System Catalog Table	Database Server Restrictions
sysblobs	None
syschecks	None
syscolauth	None
syscoldepend	None
syscolumns	None
sysconstraints	None
sysdefaults	None
sysdepend	None
sysdistrib	None
sysextcols	Available for Informix Dynamic Server with AD and XP options only
sysextdfiles	Available for Informix Dynamic Server with AD and XP options only
sysexternal	Available for Informix Dynamic Server with AD and XP options only
sysfragauth	Available for Informix Dynamic Server only
sysfragments	Not available for Dynamic Server, Workgroup and Developer Editions
sysindexes	None
sysnewdepend	Available for Informix Dynamic Server with AD and XP options only
sysobjstate	Not available for Informix Dynamic Server with AD and XP options

(1 of 2)

System Catalog Table	Database Server Restrictions
sysopclstr	Not available for Dynamic Server, Workgroup and Developer Editions
sysprocauth	None
sysprocbody	None
sysprocedures	None
sysprocplan	None
sysreferences	None
sysrepository	Available for Informix Dynamic server with AD and XP options only
sysroleauth	Not available for Dynamic Server with AD and XP options
syssynonyms	None
syssyntable	None
systabauth	None
systables	None
systrigbody	None
systriggers	None
sysusers	None
sysviews	None
sysviolations	None

(2 of 2)

Do not confuse the system catalog tables of a database with the tables in the **sysmaster** database. The **sysmaster** tables also start with the **sys** prefix, but they contain information about an entire database server, which might manage many databases. The information in the **sysmaster** tables is primarily useful for DBAs. For more information about the **sysmaster** tables, see your [Administrator's Guide](#).

In a database whose collation order is locale dependent, all character information in the system catalog tables is stored in NCHAR rather than CHAR columns. However, for those databases where the collation order is code-set dependent, all character information in the system catalog tables is stored in CHAR columns. For more information on collation orders, see the [Informix Guide to GLS Functionality](#). For information about NCHAR and CHAR data types, see the [Informix Guide to GLS Functionality](#) and Chapter 2 of this manual. ♦

SYSBLOBS

The name **sysblobs** is a legacy name based on the term blob that was used to refer to BYTE and TEXT columns.

The **sysblobs** system catalog table specifies the storage location of a BYTE or TEXT column. It contains one row for each BYTE or TEXT column in a table. The **sysblobs** system catalog table has the following columns.

Column Name	Type	Explanation
spacename	NCHAR(18)	Partition BYTE or TEXT data, dbspace, or family name
type	NCHAR(1)	Media type: M = Magnetic O = Optical. Not available for Informix Dynamic Server with AD and XP options
tabid	INTEGER	Table identifier
colno	SMALLINT	Column number

A composite index for the **tabid** and **colno** columns allows only unique values.

SYSCHECKS

The **syschecks** system catalog table describes each check constraint defined in the database. Because the **syschecks** system catalog table stores both the ASCII text and a binary encoded form of the check constraint, it contains multiple rows for each check constraint. The **syschecks** system catalog table has the following columns.

Column Name	Type	Explanation
constrid	INTEGER	Constraint identifier
type	NCHAR(1)	Form in which the check constraint is stored: B = Binary encoded T = ASCII text
seqno	SMALLINT	Line number of the check constraint
checktext	NCHAR(32)	Text of the check constraint

A composite index for the **constrid**, **type**, and **seqno** columns allows only unique values.

The text in the **checktext** column associated with B type in the **type** column is in computer-readable format. To view the text associated with a particular check constraint, use the following query with the appropriate constraint ID:

```
SELECT * FROM syschecks WHERE constrid=10 AND type='T'
```

Each check constraint described in the **syschecks** system catalog table also has its own row in the **sysconstraints** system catalog table.

SYSCOLAUTH

The **syscolauth** system catalog table describes each set of privileges granted on a column. It contains one row for each set of column privileges granted in the database. The **syscolauth** system catalog table has the following columns.

Column Name	Type	Explanation
grantor	NCHAR(8)	Grantor of privilege
grantee	NCHAR(8)	Grantee (receiver) of privilege
tabid	INTEGER	Table identifier
colno	SMALLINT	Column number
colauth	NCHAR(3)	3-byte pattern that specifies column privileges: s = Select u = Update r = References

If the **colauth** privilege code is uppercase (for example, S for select), a user who has this privilege can also grant it to others. If the **colauth** privilege code is lowercase (for example, s for select), the user who has this privilege cannot grant it to others.

A composite index for the **tabid**, **grantor**, **grantee**, and **colno** columns allows only unique values. A composite index for the **tabid** and **grantee** columns allows duplicate values.

SYSCOLDEPEND

The **syscoldepend** system catalog table tracks the table columns specified in check and not null constraints. Because a check constraint can involve more than one column in a table, the **syscoldepend** table can contain multiple rows for each check constraint. One row is created in the **syscoldepend** table for each column involved in the constraint. The **syscoldepend** system catalog table has the following columns.

Column Name	Type	Explanation
constrid	INTEGER	Constraint identifier
tabid	INTEGER	Table identifier
colno	SMALLINT	Column number

A composite index for the **constrid**, **tabid**, and **colno** columns allows only unique values. A composite index for the **tabid** and **colno** columns allows duplicate values.

SYSCOLUMNS

The **syscolumns** system catalog table describes each column in the database. One row exists for each column that is defined in a table or view.

Column Name	Type	Explanation
colname	NCHAR(18)	Column name
tabid	INTEGER	Table identifier
colno	SMALLINT	Column number that the system sequentially assigns (from left to right within each table)

(1 of 2)

Column Name	Type	Explanation
coltype	SMALLINT	Code for column data type: 0 = NCHAR 8 = MONEY 1 = SMALLINT 10 = DATETIME 2 = INTEGER 11 = BYTE 3 = FLOAT 12 = TEXT 4 = SMALLFLOAT 13 = NVARCHAR 5 = DECIMAL 14 = INTERVAL 6 = SERIAL 15 = NCHAR 7 = DATE 16 = NVARCHAR
collength	SMALLINT	Column length (in bytes)
colmin	INTEGER	Second minimum value
colmax	INTEGER	Second maximum value

(2 of 2)

A composite index for the **tabid** and **colno** columns allows only unique values.

If the **coltype** column contains a value greater than 256, it does not allow null values. To determine the data type for a **coltype** column that contains a value greater than 256, subtract 256 from the value and evaluate the remainder, based on the possible **coltype** values. For example, if a column has a **coltype** value of 262, subtracting 256 from 262 leaves a remainder of 6, which indicates that this column uses a SERIAL data type.

The value that the **collength** column holds depends on the data type of the column. If the data type of the column is BYTE or TEXT, **collength** holds the length of the descriptor. The following formula determines a **collength** value for a MONEY or DECIMAL column:

$$(precision * 256) + scale$$

For columns of type NVARCHAR, the *max_size* and *min_space* values are encoded in the **collength** column using one of the following formulas:

- If the **collength** value is positive:

$$\text{collength} = (\text{min_space} * 256) + \text{max_size}$$

- If the **collength** value is negative:

$$\text{collength} + 65536 = (\text{min_space} * 256) + \text{max_size}$$

For columns of type DATETIME or INTERVAL, the following formula determines **collength**:

$$(\text{length} * 256) + (\text{largest_qualifier_value} * 16) + \text{smallest_qualifier_value}$$

The *length* is the physical length of the DATETIME or INTERVAL field, and *largest_qualifier* and *smallest_qualifier* have the values that the following table shows.

Field Qualifier	Value
YEAR	0
MONTH	2
DAY	4
HOUR	6
MINUTE	8
SECOND	10
FRACTION(1)	11
FRACTION(2)	12
FRACTION(3)	13
FRACTION(4)	14
FRACTION(5)	15

For example, if a DATETIME YEAR TO MINUTE column has a length of 12 (such as YYYY:DD:MM:HH:MM), a *largest_qualifier* value of 0 (for YEAR), and a *smallest_qualifier* value of 8 (for MINUTE), the **collength** value is 3080 or $(256 * 12) + (0 * 16) + 8$.

For information about how to use the HEX function to display the **collength** and **coltype** values, see the Column Expression discussion in the Expression segment of the *Informix Guide to SQL: Syntax*.

The **colmin** and **colmax** column values hold the second-smallest and second-largest data values in the column, respectively. For example, if the values in an indexed column are 1, 2, 3, 4, and 5, the **colmin** value is 2 and the **colmax** value is 4. Storing the second-smallest and second-largest data values lets the database server make assumptions about the range of values in a given column and, in turn, further optimize searching strategies. The **colmin** and **colmax** columns contain values only if the column is indexed and you have run the UPDATE STATISTICS statement. If you store BYTE or TEXT data in the **tblspace**, the **colmin** value is -1. The values for all other noninteger column types are the initial 4 bytes of the maximum or minimum value, which are treated as an integer.

SYSCONSTRAINTS

The **sysconstraints** system catalog table lists the constraints placed on the columns in each database table. An entry is also placed in the **sysindexes** system catalog table for each unique, primary key, or referential constraint that you create, if the constraint does not already have a corresponding entry in the **sysindexes** system catalog table. Because indexes can be shared, more than one constraint can be associated with an index. The **sysconstraints** system catalog table has the following columns.

Column Name	Type	Explanation
constrid	SERIAL	System-assigned sequential identifier
constrname	NCHAR(18)	Constraint name
owner	NCHAR(8)	User name of owner
tabid	INTEGER	Table identifier

(1 of 2)

Column Name	Type	Explanation
constrtype	NCHAR(1)	Constraint type: C = Check constraint P = Primary key R = Referential U = Unique N = Not null
idxname	NCHAR(18)	Index name

(2 of 2)

A composite index for the **constname** and **owner** columns allows only unique values. The index for the **tabid** column allows duplicate values, and the index for the **constrid** column allows only unique values.

For check constraints (where **constrtype** = C), the **idxname** is always null. Additional information about each check constraint is contained in the **syschecks** system catalog table.

SYSDEFAULTS

The **sysdefaults** system catalog table lists the user-defined defaults that are placed on each column in the database. One row exists for each user-defined default value. If a default is not explicitly specified in the CREATE TABLE statement, no entry exists in this table. The **sysdefaults** system catalog table has the following columns.

Column Name	Type	Explanation
tabid	INTEGER	Table identifier
colno	SMALLINT	Column identifier

(1 of 2)

Column Name	Type	Explanation
type	NCHAR(1)	Default type: L = Literal default U = User C = Current N = Null T = Today S = Dbservername
default	NCHAR(256)	If default type = L, the literal default value

(2 of 2)

If you specify a literal for the default value, it is stored in the **default** column as ASCII text. If the literal value is not of type NCHAR, the **default** column consists of two parts. The first part is the 6-bit representation of the binary value of the default value structure. The second part is the default value in English text. A space separates the two parts.

If the data type of the column is not NCHAR or NVARCHAR, a binary representation is encoded in the **default** column.

A composite index for both the **tabid** and **colno** columns allows only unique values.

SYSDEPEND

The **sysdepend** system catalog table describes how each view or table depends on other views or tables. One row exists in this table for each dependency, so a view based on three tables has three rows. The **sysdepend** system catalog table has the following columns.

Column Name	Type	Explanation
btoid	INTEGER	Table identifier of base table or view
btype	NCHAR(1)	Base object type: T = Table V = View
doid	INTEGER	Table identifier of dependent table
dtype	NCHAR(1)	Dependent object type (V = View); currently, only view is implemented

The **btoid** and **doid** columns are indexed and allow duplicate values.

SYSDISTRIB

The **sysdistrib** system catalog table stores data-distribution information for the database server to use. Data distributions provide detailed table-column information to the optimizer to improve the choice of execution paths of SQL SELECT statements. Information is stored in the **sysdistrib** system catalog table when an UPDATE STATISTICS statement with mode MEDIUM or HIGH is run for a table. The **sysdistrib** system catalog table has the following columns.

Column Name	Type	Explanation
tabid	INTEGER	Table identifier of the table where data was gathered
colno	SMALLINT	Column number in the source table
seqno	INTEGER	Sequence number for multiple entries
constructed	DATE	Date when the data distribution was created
mode	NCHAR(1)	Optimization level: L = Low M = Medium H = High
resolution	FLOAT	Specified in the UPDATE STATISTICS statement
confidence	FLOAT	Specified in the UPDATE STATISTICS statement
encdat	NCHAR(256)	ASCII-encoded histogram in fixed-length character field; accessible only to user informix

You can select any column from **sysdistrib** except **encdat**. User **informix** can select the **encdat** column.

SYSEXTCOLS

This table is available for Dynamic Server with AD and XP Options only. The **sysextcols** system catalog table contains a row that describes each of the internal columns in external table **tabid** of format type (fmttype) **FIXED**. No entries are stored in **sysextcols** for **DELIMITED** or Informix-format external files.

Column	Type	Description
tabid	integer	Table identifier
colno	smallint	Column identifier
exttype	smallint	External column type
extstart	smallint	Starting position of column in the external data file
extlength	smallint	External column length in bytes
nullstr	char(256)	Represents null in external data
picture	char(256)	Reserved for future use
decimal	smallint	Precision for external decimals
extstype	char(18)	The external type name

You can use **DBSCHEMA** to write out the description of the external tables. To query these catalogs about an external table, use the **tabid** as stored in **systables** with **tabtype** = 'E'.

AD/XP

SYSEXTDFILES

This table is available for Dynamic Server with AD and XP Options only. For each external table, at least one row exists in the **sysextfiles** system catalog table.

Column	Type	Description
tabid	integer	Table identifier
dfentry	char(152)	Data file entry

You can use DBSCHEMA to write out the description of the external tables. To query these system catalogs about an external table, use the tabid as stored in **systables** with tabtype = 'E'.

AD/XP

SYSEXTERNAL

This table is available for Dynamic Server with AD and XP Options only. For each external table, a single row exists in the **sysexternal** system catalog table. The **tabid** column associates the external table in this system catalog table with an entry in **systables**.

Column	Type	Description
tabid	integer	Table identifier
fnttype	char(1)	'D' (delimiter), 'F' (fixed), 'I' (Informix)
recdelim	char(4)	The record delimiter
flddelim	char(4)	The field delimiter
codeset	char(18)	ASCII, EBCDIC
datefmt	char(8)	Reserved for future use
moneyfmt	char(20)	Reserved for future use
maxerrors	integer	Number of errors to allow per coserver

(1 of 2)

Column	Type	Description
rejectfile	char(128)	Name of reject file
flags	integer	Optional load flags
ndfiles	integer	Number of data files in sysextdfiles

(2 of 2)

You can use DBSCHEMA to write out the description of the external tables. To query these catalogs about an external table, use the tabid as stored in **systables** with tabtype = 'E'.

SYSFRAGAUTH

The **sysfragauth** system catalog table is available only for Dynamic Server. The **sysfragauth** system catalog table stores information about the privileges that are granted on table fragments. The **sysfragauth** system catalog table has the following columns.

Column Name	Type	Explanation
grantor	NCHAR(8)	Grantor of privilege
grantee	NCHAR(8)	Grantee (receiver) of privilege
tabid	INTEGER	Table identifier of the table that contains the fragment named in the fragment column.
fragment	NCHAR(18)	Name of dbspace where fragment is stored. Identifies the fragment on which privileges are granted.
fragauth	NCHAR(6)	A 6-byte pattern that specifies fragment-level privileges (including 3 bytes reserved for future use). This pattern contains one or more of the following codes: u = Update i = Insert d = Delete

W/D

The **sysfragauth** system catalog table is not available for Dynamic Server, Workgroup and Developer Editions. ♦

If a code in the **fragauth** column is lowercase, the grantee cannot grant the privilege to other users. If a code in the **fragauth** column is uppercase, the grantee can grant the privilege to other users.

A composite index for the **tabid**, **grantor**, **grantee**, and **fragment** columns allows only unique values. A composite index on the **tabid** and **grantee** columns allows duplicate values.

The following example displays the fragment-level privileges for one base table, as they appear in the **sysfragauth** system catalog table. The grantee **ted** can grant the UPDATE, DELETE, and INSERT privileges to other users.

grantor	grantee	tabid	fragment	fragauth
dba	dick	101	dbsp1	-ui---
dba	jane	101	dbsp3	--i---
dba	mary	101	dbsp4	--id--
dba	ted	101	dbsp2	-UID--

W/D

SYSFRAGMENTS

The **sysfragments** system catalog table is not available for Dynamic Server, Workgroup and Developer Editions. ♦

The **sysfragments** system catalog table stores fragmentation information for tables and indexes. One row exists for each table or index fragment. The **sysfragments** table has the following columns.

Column Name	Type	Explanation
fragtype	NCHAR(1)	Fragment type: I = Index T = Table B = TEXT or BYTE data (Informix Dynamic Server with AD and XP options only)
tabid	INTEGER	Table identifier
indexname	NCHAR(18)	Index identifier
colno	SMALLINT	TEXT or BYTE column identifier
partn	INTEGER	Physical location identifier
strategy	NCHAR(1)	Distribution scheme type: R = Round-robin strategy was used to distribute the fragments E = Expression-based strategy was used to distribute the fragments T = Table-based strategy was used to distribute the fragments I = IN DBSPACE clause specified a specific location as part of the fragmentation strategy H = hash-based strategy was used to distribute the fragments (Informix Dynamic Server with AD and XP options only)
location	NCHAR(1)	Reserved for future use; shows L for local
servername	NCHAR(18)	Reserved for future use

(1 of 2)

Column Name	Type	Explanation
evalpos	INTEGER	Position of fragment in the fragmentation list
exprtext	TEXT	Expression that was entered for fragmentation strategy Names of the columns that are hashed. (Informix Dynamic Server with AD and XP options only)
exprbin	BYTE	Binary version of expression
exprarr	BYTE	Range partitioning data used to optimize expression in range-expression fragmentation strategy
flags	INTEGER	Internally used
dbspace	NCHAR(18)	Dbspacename for fragment
levels	SMALLINT	Number of B+ tree index levels
npused	INTEGER	For table fragmentation strategy, npused represents the number of data pages; for index fragmentation strategy, npused represents the number of leaf pages.
nrows	INTEGER	For tables, nrows represents the number of rows in the fragment; for indexes, nrows represents the number of unique keys.
clust	INTEGER	Degree of index clustering; smaller numbers correspond to greater clustering
hybdpos	INTEGER	Contains the relative position of the fragment within the dbslice. The hybrid fragmentation strategy and the set of fragments against which the hybrid strategy is applied determines the relative position. The first fragment has a hybdpos value of zero. (Informix Dynamic Server with AD and XP options only)

(2 of 2)

The strategy type **T** is used for attached indexes (where index fragmentation is the same as the table fragmentation).

SYSINDEXES

The **sysindexes** system catalog table describes the indexes in the database. It contains one row for each index that is defined in the database. The **sysindexes** system catalog table has the following columns.

Column Name	Type	Explanation
idxname	NCHAR (18)	Index name
owner	NCHAR(8)	Owner of index (user informix for system catalog tables and user name for database tables)
tabid	INTEGER	Table identifier
idxtype	NCHAR(1)	Index type: U = Unique G = Non-bitmap general-key index (Informix Dynamic Server with AD and XP options only) D = Duplicates g = Bitmap general-key index (Informix Dynamic Server with AD and XP options only) u = unique, bitmap (Informix Dynamic Server with AD and XP options only) d = non-unique, bitmap (Informix Dynamic Server with AD and XP options only)
clustered	NCHAR(1)	Clustered or nonclustered index (C = Clustered)
part1	SMALLINT	Column number (colno) of a single index or the 1st component of a composite index
part2	SMALLINT	2nd component of a composite index
part3	SMALLINT	3rd component of a composite index
part4	SMALLINT	4th component of a composite index

(1 of 2)

Column Name	Type	Explanation
part5	SMALLINT	5th component of a composite index
part6	SMALLINT	6th component of a composite index
part7	SMALLINT	7th component of a composite index
part8	SMALLINT	8th component of a composite index
part9	SMALLINT	9th component of a composite index
part10	SMALLINT	10th component of a composite index
part11	SMALLINT	11th component of a composite index
part12	SMALLINT	12th component of a composite index
part13	SMALLINT	13th component of a composite index
part14	SMALLINT	14th component of a composite index
part15	SMALLINT	15th component of a composite index
part16	SMALLINT	16th component of a composite index
levels	SMALLINT	Number of B+ tree levels
leaves	INTEGER	Number of leaves
nunique	INTEGER	Number of unique keys in the first column
clust	INTEGER	Degree of clustering; smaller numbers correspond to greater clustering

(2 of 2)

Changes that affect existing indexes are reflected in this table only after you run the UPDATE STATISTICS statement.

Each **part n th** column component of a composite index (the **part1** through **part16** columns in this table) holds the column number (**colno**) of each part of the 16 possible parts of a composite index. If the component is ordered in descending order, the **colno** is entered as a negative value.

The **clust** column is blank until the UPDATE STATISTICS statement is run on the table. The maximum value is the number of rows in the table, and the minimum value is the number of data pages in the table.

The **tabid** column is indexed and allows duplicate values. A composite index for the **idxname**, **owner**, and **tabid** columns allows only unique values.

SYSNEWDEPEND

The **sysnewdepend** system catalog table is available only for Dynamic Server with AD and XP Options.

The **sysnewdepend** system catalog table contains information about general-key indexes that is not available in the **sysindexes** system catalog table. The dependencies between a general-key index and the tables in the FROM clause of the CREATE INDEX statement are stored in the **sysnewdepend** system catalog table. The **sysnewdepend** system catalog table has the following columns.

Column Name	Type	Explanation
scrid1	CHAR	The name of the general-key index
scrid2	INTEGER	The tableid of the indexed table
type	INTEGER	The type of general-key index
destid1	INTEGER	The tableid of the table that the general-key index depends.
destid2	INTEGER	The column name in the table that the general-key index depends.

SYSOBJSTATE

The **sysobjstate** system catalog table is available only for Dynamic Server. The **sysobjstate** system catalog table stores information about the state (object mode) of database objects. The types of database objects listed in this table are indexes, triggers, and constraints.

Every index, trigger, and constraint in the database has a corresponding row in the **sysobjstate** table if a user creates the object. Indexes that the database server creates on the system catalog tables are not listed in the **sysobjstate** table because their object mode cannot be changed.

The **sysobjstate** system catalog table has the following columns.

Column Name	Type	Explanation
objtype	NCHAR(1)	The type of database object. This column has one of the following codes: C = Constraint I = Index T = Trigger
owner	NCHAR(8)	The owner of the database object
name	NCHAR(18)	The name of the database object
tabid	INTEGER	Table identifier of the table on which the database object is defined
state	NCHAR(1)	The current state (object mode) of the database object. This column has one of the following codes: D = Disabled E = Enabled F = Filtering with no integrity-violation errors G = Filtering with integrity-violation errors

A composite index for the **objtype**, **name**, **owner**, and **tabid** columns allows only unique values.

IDS

SYSOPCLSTR

The **sysopclstr** system catalog table is available only for Dynamic Server. The **sysopclstr** system catalog table defines each optical cluster in the database. It contains one row for each optical cluster. The **sysopclstr** system catalog table has the following columns.

W/D

The **sysopclstr** system catalog table is not available for Dynamic Server, Workgroup and Developer Editions. ♦

Column Name	Type	Explanation
owner	NCHAR(8)	Owner of the cluster
clstrname	NCHAR(18)	Name of the cluster
clstrsize	INTEGER	Size of the cluster
tabid	INTEGER	Table identifier
blobcol1	SMALLINT	BYTE or TEXT column number 1
blobcol2	SMALLINT	BYTE or TEXT column number 2
blobcol3	SMALLINT	BYTE or TEXT column number 3
blobcol4	SMALLINT	BYTE or TEXT column number 4
blobcol5	SMALLINT	BYTE or TEXT column number 5
blobcol6	SMALLINT	BYTE or TEXT column number 6
blobcol7	SMALLINT	BYTE or TEXT column number 7
blobcol8	SMALLINT	BYTE or TEXT column number 8
blobcol9	SMALLINT	BYTE or TEXT column number 9
blobcol10	SMALLINT	BYTE or TEXT column number 10
blobcol11	SMALLINT	BYTE or TEXT column number 11
blobcol12	SMALLINT	BYTE or TEXT column number 12
blobcol13	SMALLINT	BYTE or TEXT column number 13
blobcol14	SMALLINT	BYTE or TEXT column number 14
blobcol15	SMALLINT	BYTE or TEXT column number 15
blobcol16	SMALLINT	BYTE or TEXT column number 16
clstrkey1	SMALLINT	Cluster key number 1
clstrkey2	SMALLINT	Cluster key number 2

(1 of 2)

Column Name	Type	Explanation
clstrkey3	SMALLINT	Cluster key number 3
clstrkey4	SMALLINT	Cluster key number 4
clstrkey5	SMALLINT	Cluster key number 5
clstrkey6	SMALLINT	Cluster key number 6
clstrkey7	SMALLINT	Cluster key number 7
clstrkey8	SMALLINT	Cluster key number 8
clstrkey9	SMALLINT	Cluster key number 9
clstrkey10	SMALLINT	Cluster key number 10
clstrkey11	SMALLINT	Cluster key number 11
clstrkey12	SMALLINT	Cluster key number 12
clstrkey13	SMALLINT	Cluster key number 13
clstrkey14	SMALLINT	Cluster key number 14
clstrkey15	SMALLINT	Cluster key number 15
clstrkey16	SMALLINT	Cluster key number 16

(2 of 2)

A composite index for both the **clstrname** and **owner** columns allows only unique values. The **tabid** column allows duplicate values.

SYSPROCAUTH

The **sysprocauth** system catalog table describes the privileges granted on a procedure. It contains one row for each set of privileges that are granted. The **sysprocauth** system catalog table has the following columns.

Column Name	Type	Explanation
grantor	NCHAR(8)	Grantor of procedure
grantee	NCHAR(8)	Grantee (receiver) of procedure
procid	INTEGER	Procedure identifier
procauth	NCHAR(1)	Type of procedure permission granted: e = Execute permission on procedure E = Execute permission and the ability to grant it to others

A composite index for the **procid**, **grantor**, and **grantee** columns allows only unique values. The composite index for the **procid** and **grantee** columns allows duplicate values.

SYSPROCBODY

The **sysprocbody** system catalog table describes the compiled version of each stored procedure in the database. Because the **sysprocbody** system catalog table stores the text of the procedure, each procedure can have multiple rows. The **sysprocbody** system catalog table has the following columns.

Column Name	Type	Explanation
procid	INTEGER	Procedure identifier
datakey	NCHAR(1)	Data-descriptor type: D = User document text T = Actual procedure source R = Return value type list S = Procedure symbol table L = Constant procedure data string (that is, literal numbers or quoted strings) P = Interpreter instruction code
seqno	INTEGER	Line number of the procedure
data	NCHAR(256)	Actual text of the procedure

Although the **datakey** column indicates the type of data that is stored, the **data** column contains the actual data, which can be one of the following types:

- Encoded return values list
- Encoded symbol table
- Constant data
- Compiled code for the procedure
- Text of the procedure and its documentation

A composite index for the **procid**, **datakey**, and **seqno** columns allows only unique values.

SYSPROCEDURES

The **sysprocedures** system catalog table lists the characteristics for each stored procedure in the database. It contains one row for each procedure. The **sysprocedures** system catalog table has the following columns.

Column Name	Type	Explanation
procname	NCHAR(18)	Procedure name
owner	NCHAR(8)	Owner name
procid	SERIAL	Procedure identifier
mode	NCHAR(1)	Mode type: D = DBA O = Owner P = Protected
retsize	INTEGER	Compiled size (in bytes) of values
symsize	INTEGER	Compiled size (in bytes) of symbol table
datasize	INTEGER	Compiled size (in bytes) constant data
codesize	INTEGER	Compiled size (in bytes) of procedure instruction code
numargs	INTEGER	Number of procedure arguments

A composite index for the **procname** and **owner** columns allows only unique values.

A database server can create protected stored procedures for internal use. The **sysprocedures** table identifies these protected stored procedures with the letter **P** in the mode column. You cannot modify or drop protected stored procedures or display them through **dbschema**.

SYSPROCPLAN

The **sysprocplan** system catalog table describes the query-execution plans and dependency lists for data-manipulation statements within each stored procedure. If new plans are generated during the execution of a stored procedure, the new plans are also recorded in **sysprocplan**. Because different parts of a procedure plan can be created on different dates, the table can contain multiple rows for each procedure.

It is possible to delete all the plans for a particular stored procedure with the DELETE statement on **sysprocplan**. When the procedure is executed, new plans are automatically generated and recorded in **sysprocplan**.

If you are using Dynamic Server with AD and XP Options, and you delete plans from **sysprocplan** and execute the procedure from any coserver, then the plans are not stored in **sysprocplan**. Issue UPDATE STATISTICS FOR PROCEDURE from any coserver to update the plans in **sysprocplan**. ♦

The **sysprocplan** system catalog table has the following columns.

Column Name	Type	Explanation
procid	INTEGER	Procedure identifier
planid	INTEGER	Plan identifier
datakey	NCHAR(1)	Identifier procedure plan part: D = Dependency list Q = Execution plan
seqno	INTEGER	Line number of plan
created	DATE	Date plan created
datasize	INTEGER	Size (in bytes) of the list or plan
data	NCHAR(256)	Encoded (compiled) list or plan

A composite index for the **procid**, **planid**, **datakey**, and **seqno** columns allows only unique values.

SYSREFERENCES

The **sysreferences** system catalog table lists the referential constraints that are placed on columns in the database. It contains a row for each referential constraint in the database. The **sysreferences** system catalog table has the following columns.

Column Name	Type	Explanation
constrid	INTEGER	Constraint identifier
primary	INTEGER	Constraint identifier of the corresponding primary key
ptabid	INTEGER	Table identifier of the primary key
updrule	NCHAR(1)	Reserved for future use; displays an R
delrule	NCHAR(1)	Displays cascading delete or restrict rule: C = Cascading delete R = Restrict (default)
matchtype	NCHAR(1)	Reserved for future use; displays an N
pendant	NCHAR(1)	Reserved for future use; displays an N

The **constrid** column is indexed and allows only unique values. The **primary** column is indexed and allows duplicate values.

SYSREPOSITORY

The **sysrepository** system catalog table contains information about generalized-key indexes that is not available in the **sysindexes** system catalog table. The **sysrepository** system catalog table contains the CREATE statement for each generalized-key index in the database in its **desc** column. The contents of the **sysrepository** system catalog table are useful when a generalized-key index has to be rebuilt during a recovery or if a user wants to see the CREATE statement for a specific generalized-key index.

Column Name	Type	Explanation
id1	CHAR	Index from the generalized-key index
id2	INTEGER	Tabid of table with the generalized-key index
type	INTEGER	Integer representing object type In this release, the only integer that shows is 1, indicating generalized-key index type.
seqid	INTEGER	For future use
desc	TEXT	The CREATE statement used for each general-key index in the database
bin	BYTE	Internal representation of the generalized-key index

SYSROLEAUTH

The **sysroleauth** system catalog table describes the roles that are granted to users. It contains one row for each role that is granted to a user in the database. The **sysroleauth** system catalog table has the following columns.

Column Name	Type	Explanation
rolename	NCHAR(8)	Name of the role
grantee	NCHAR(8)	Grantee (receiver) of role
is_grantable	NCHAR(1)	Specifies whether the role is grantable: Y = Grantable N = Not grantable

The **rolename** and **grantee** columns are indexed and allow only unique values. The **is_grantable** column indicates whether the role was granted with the WITH GRANT OPTION on the GRANT statement.

SYSSYNONYMS

The **syssynonyms** system catalog table lists the synonyms for each table or view. It contains a row for every synonym defined in the database. The **syssynonyms** system catalog table has the following columns.

Column Name	Type	Explanation
owner	NCHAR(8)	User name of owner
synname	NCHAR(18)	Synonym identifier
created	DATE	Date synonym created
tabid	INTEGER	Table identifier

A composite index for the **owner** and **synonym** columns allows only unique values. The **tabid** column is indexed and allows duplicate values.

***Important:** Version 4.0 or later Informix products no longer use this table; however, any **syssynonyms** entries made before Version 4.0 remain in this table.*

SYSSYNTABLE

The **syssyntable** system catalog table outlines the mapping between each synonym and the object that it represents. It contains one row for each entry in the **systables** table that has a **tabtype** of **S**. The **syssyntable** system catalog table has the following columns.

Column Name	Type	Explanation
tabid	INTEGER	Table identifier
servername	NCHAR(18)	Server name
dbname	NCHAR(18)	Database name

(1 of 2)



Column Name	Type	Explanation
owner	NCHAR(8)	User name of owner
tabname	NCHAR(18)	Name of table
tabid	INTEGER	Table identifier of base table or view

(2 of 2)

If you define a synonym for a table that is in your current database, only the **tabid** and **btabid** columns are used. If you define a synonym for a table that is external to your current database, the **btabid** column is not used, but the **tabid**, **servername**, **dbname**, **owner**, and **tabname** columns are used.

The **tabid** column maps to the **tabid** column in **systables**. With the **tabid** information, you can determine additional facts about the synonym from **systables**.

An index for the **tabid** column allows only unique values. The **btabid** column is indexed to allow duplicate values.

SYSTABAUTH

The **systabauth** system catalog table describes each set of privileges that are granted in a table. It contains one row for each set of table privileges that are granted in the database. The **systabauth** system catalog table has the following columns.

Column Name	Type	Explanation
grantor	NCHAR(8)	Grantor of privilege
grantee	NCHAR(8)	Grantee (receiver) of privilege
tabid	INTEGER	Table identifier

(1 of 2)

Column Name	Type	Explanation
tabauth	NCHAR(8)	8-byte pattern that specifies table privileges: s = Select u = Update * = Column-level authority i = Insert d = Delete x = Index a = Alter r = References

(2 of 2)

If the **tabauth** privilege code is uppercase (for example, S for select), a user who has this privilege can grant it to others. If the **tabauth** privilege code is lowercase (for example, s for select), the user who has this privilege cannot grant it to others.

A composite index for the **tabid**, **grantor**, and **grantee** columns allows only unique values. The composite index for the **tabid** and **grantee** columns allows duplicate values.

SYSTABLES

The **systables** system catalog table describes each table in the database. It contains one row for each table, view, or synonym that is defined in the database. The information in the **systables** system catalog table includes all database tables and the system catalog tables. The **systables** system catalog table has the following columns

Column Name	Type	Explanation
tablename	NCHAR(18)	Name of table, view, or synonym
owner	NCHAR(8)	Owner of table (user informix for system catalog tables and user name for database tables)
partnum	INTEGER	Tblspace identifier (similar to tabid)
tabid	SERIAL	System-assigned sequential ID number (system tables: 1-24, user tables: 100-nnn)
rowsize	SMALLINT	Row size
ncols	SMALLINT	Number of columns
nindexes	SMALLINT	Number of indexes
nrows	INTEGER	Number of rows
created	DATE	Date created
version	INTEGER	Number that changes when table is altered
tabtype	NCHAR(1)	Table type: T = Table V = View P = Private synonym P = Synonym (in an ANSI-compliant database) S = Synonym
locklevel	NCHAR(1)	Lock mode for a table: B = Page P = Page R = Row T = Table - OnLine Informix Dynamic Server with AD and XP options only
npused	INTEGER	Number of data pages in use

(1 of 2)

Column Name	Type	Explanation
fextsize	INTEGER	Size of initial extent (in kilobytes)
nextsize	INTEGER	Size of all subsequent extents (in kilobytes)
flags	SMALLINT	Has a unique value for the four types of permanent tables: RAW 0x00000002 Informix Dynamic Server with AD and XP options only STATIC 0x00000004 Informix Dynamic Server with AD and XP options only OPERATIONAL 0x00000010 Informix Dynamic Server with AD and XP options only STANDARD 0x00000020 Informix Dynamic Server with AD and XP options only EXTERNAL 0x00000020 Informix Dynamic Server with AD and XP options only
site	NCHAR(18)	Reserved for future use (used to store database collation and C-type information)
dbname	NCHAR(18)	Reserved for future use

(2 of 2)

The **tabid** column is indexed and must contain unique values. A composite index for the **tablename** and **owner** columns allows only unique values. The **version** column contains an encoded number that is put in the **systables** system catalog table when the table is created. Portions of the encoded value are incremented when data-definition statements, such as ALTER INDEX, ALTER TABLE, DROP INDEX, and CREATE INDEX, are performed. When a prepared statement is executed, the **version** number is checked to make sure that nothing has changed since the statement was prepared. If the **version** number has changed, your statement does not execute, and you must prepare your statement again.

The **npused** column does not reflect BYTE or TEXT data used.

The **systables** system catalog table has two additional rows to store the database locale: GL_COLLATE with a **tabid** of 90 and GL_CTYPE with a **tabid** of 91. Enter the following SELECT statement to view these rows:

```
SELECT tabname, tabid FROM systables
```

SYSTRIGBODY

The **systrigbody** system catalog table contains the English text of the trigger definition and the linearized code for the trigger. Linearized code is binary data and code that is represented in ASCII format.

Warning: The database server uses the linearized code that is stored in **systrigbody**. You must not alter the content of rows that contain linearized code.

The **systrigbody** system catalog table has the following columns.

Column Name	Type	Explanation
trigid	INT	Trigger identifier
datakey	NCHAR	Type of data: D = English text for the header, trigger definition A = English text for the body, triggered actions H = Linearized code for the header S = Linearized code for the symbol table B = Linearized code for the body
seqno	INT	Sequence number
data	NCHAR(256)	English text or linearized code

A composite index for the **trigid**, **datakey**, and **seqno** columns allows only unique values.



SYSTRIGGERS

The **systriggers** system catalog table contains miscellaneous information about the SQL triggers in the database. This information includes the trigger event and the correlated reference specification for the trigger. The **systriggers** system catalog table has the following columns.

Column Name	Type	Explanation
trigid	SERIAL	Trigger identifier
trigname	NCHAR(18)	Trigger name
owner	NCHAR(8)	Owner of trigger
tabid	INT	ID of triggering table
event	NCHAR	Triggering event: I = Insert trigger U = Update trigger D = Delete trigger
old	NCHAR(18)	Name of value before update
new	NCHAR(18)	Name of value after update
mode	NCHAR	Reserved for future use

A composite index for the **trigname** and **owner** columns allows only unique values. The **trigid** column is indexed and must contain unique values. An index for the **tabid** column allows duplicate values.

SYSUSERS

The **sysusers** system catalog table describes each set of privileges that are granted in the database. It contains one row for each user who has privileges in the database. The **sysusers** system catalog table has the following columns.

Column Name	Type	Explanation
username	NCHAR(8)	Name of the database user or role
usertype	NCHAR(1)	Specifies database-level privileges: D = DBA (all privileges) R = Resource (create permanent tables and indexes) G = Role C = Connect (work within existing tables)
priority	SMALLINT	Reserved for future use
password	CHAR(8)	Reserved for future use

The **username** column is indexed and allows only unique values. The **username** can be the name of a role.

SYSVIEWS

The **sysviews** system catalog table describes each view that is defined in the database. Because the **sysviews** system catalog table stores the SELECT statement that you use to create the view, it can contain multiple rows for each view in the database. The **sysviews** system catalog table has the following columns.

Column Name	Type	Explanation
tabid	INTEGER	Table identifier
seqno	SMALLINT	Line number of the SELECT statement
viewtext	NCHAR(64)	Actual SELECT statement used to create the view

A composite index for the **tabid** and **seqno** columns allows only unique values.

SYSVIOLATIONS

The **sysviolations** system catalog table stores information about the constraint violations for base tables. Every table in the database that has a violations table and a diagnostics table associated with it has a corresponding row in the **sysviolations** table. The **sysviolations** system catalog table has the following columns.

Column Name	Type	Explanation
targettid	INTEGER	Table identifier of the target table. The target table is the base table on which the violations table and the diagnostic table are defined.
viotid	INTEGER	Table identifier of the violations table
diatid	INTEGER	Table identifier of the diagnostics table

(1 of 2)

Column Name	Type	Explanation
maxrows	INTEGER	Maximum number of rows that can be inserted in the diagnostics table during a single insert, update, or delete operation on a target table that has a filtering mode object defined on it. Also signifies the maximum number of rows that can be inserted in the diagnostics table during a single operation that enables a disabled object or sets a disabled object to filtering mode (provided that a diagnostics table exists for the target table). If no maximum is specified for the diagnostics table, this column contains a null value.

(2 of 2)

The primary key of the **sysviolations** table is the **targettid** column. Unique indexes are also defined on the **viotid** and **diatid** columns.

Dynamic Server with AD and XP Options does not use the diagnostic table when a constraint violation occurs. Rather, Dynamic Server with AD and XP Options stores additional information in the violations table. The violations table contains the data that was refused by the transaction and an indication of the cause. ♦

System Catalog Map

Figure 1-1 displays the table names and column names of the tables in a system catalog. The grey bars that connect a column in one table to a column in another table indicate columns that contain the same information.

Dynamic Server with AD and XP Options not only uses the system catalog tables that Figure 1-1 shows, but it also generates and uses the additional system catalog tables that Figure 1-2 shows. ♦

Figure 1-1
System Catalog Map

systables

tabid tabname owner nrowsize ncols nindexes nrow created version tabtype
partnum locklevel npused fextsize nextsize flags site dbname

syssttable

tabid tabname servername dbname owner btabid

sysstsynonyms

tabid owner synname created

sysdepend

btabid btype dtabid dtype

syscolumns

tabid colno colname coltype collength colmin colmax

syscolauth

tabid colno grantor grantee colauth

sysusers

username usertype priority password

sysindexes

tabid idxname owner idxtype clustered part1-part16 levels leaves nunique clust

sysconstraints

tabid idxname constrid constrname owner constrtype

sysfragments

tabid fragtype indexname colno partn strategy location servername evalpos
exprrtext exprrbin exprrarr flags dbspace levels npused nrow clust

sysopclstr

tabid owner clstrname clstrsize blobcol1-blobcol16 clstrkey1-clstrkey16

systriggers

tabid trigid trigname owner event old new mode

sysstrigbody

trigid datakey seqno data

sysdistrib

tabid colno seqno constructed mode resolution confidence encdat

systabauth

tabid grantor grantee tabauth

sysviews

tabid seqno viewtext

sysblobs

tabid spacename type colno

sysfragauth

tabid grantor grantee fragment fragauth

sysobjstate

tabid objtype owner name state

sysprocplan

procid planid datakey seqno created
datasize data

sysprocbody

procid datakey seqno data

sysprocedures

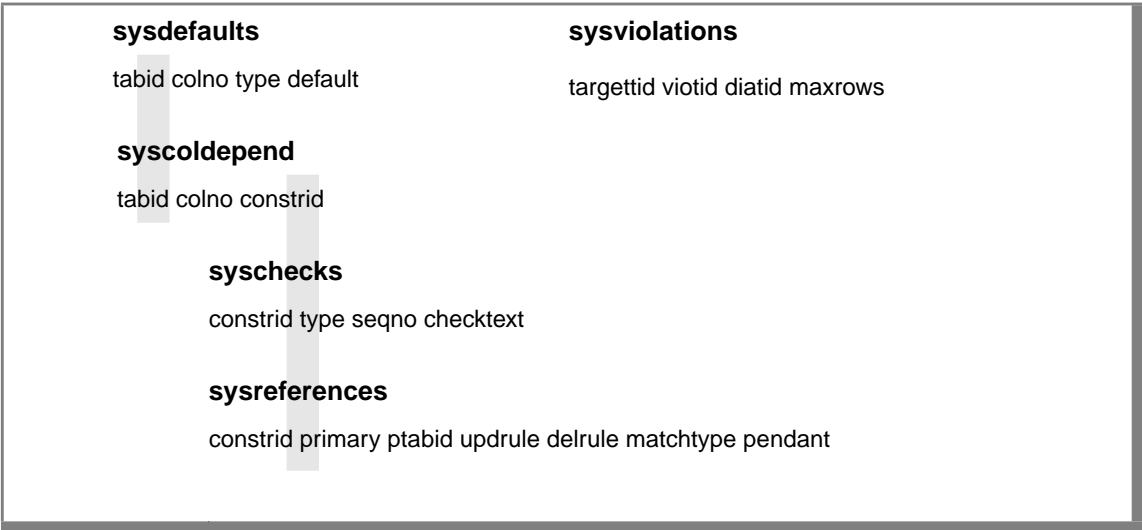
procid procname owner mode retsize
symsize datasize codesize numargs

sysprocauth

procid procauth grantor grantee

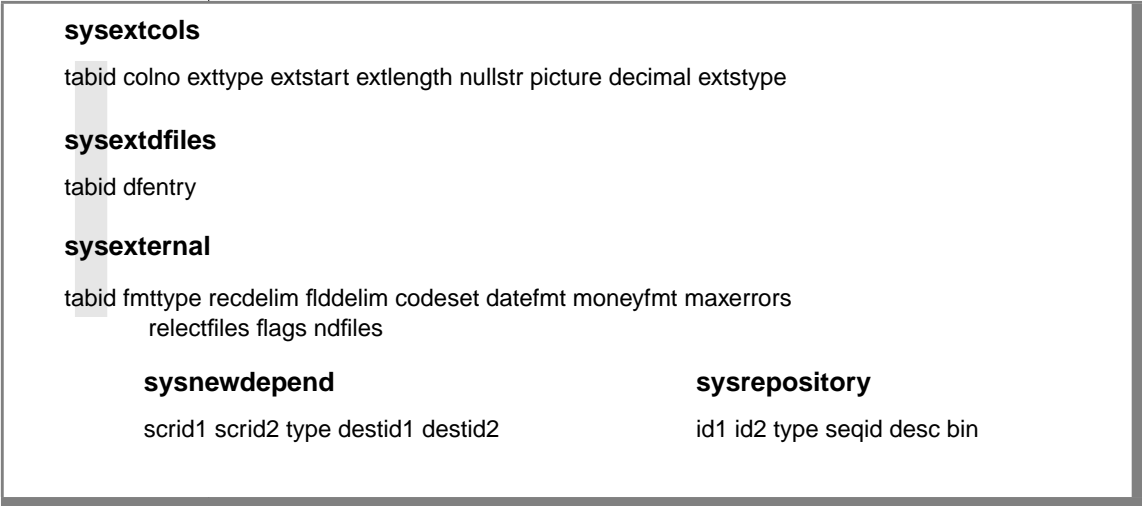
sysroleauth

rolename is_grantable grantee



(3 of 3)

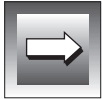
Figure 1-2
Additional System Catalogs for the
AD/XP option



Information Schema

The Information Schema consists of read-only views that provide information about all the tables, views, and columns on the current database server to which you have access. In addition, Information Schema views provide information about SQL dialects (such as Informix, Oracle, or Sybase) and SQL standards.

This version of the Information Schema views are X/Open CAE standards. Informix provides them so that applications developed on other database systems can obtain Informix system catalog information without having to use the Informix system catalogs directly.



Important: *Because the X/Open CAE standards Information Schema views differ from ANSI-compliant Information Schema views, Informix recommends that you do not install the X/Open CAE Information Schema views on ANSI-compliant databases.*

The following Information Schema views are available:

- **tables**
- **columns**
- **sql_languages**
- **server_info**

The following sections contain information about how to generate and access Information Schema views as well as information about their structure.

Generating the Information Schema Views

The Information Schema views are generated automatically when you, as DBA, run the following DB-Access command:

```
dbaccess database-name $INFORMIXDIR/etc/xpg4_is.sql
```

Data in the Informix system catalog tables populates the views. If tables, views, or stored procedures exist with any of the same names as the Information Schema views, you need to either rename the database objects or rename the views in the script before you can install the views. You can drop the views with the DROP VIEW statement on each view. To re-create the views, rerun the script.



Important: In addition to the columns specified for each Information Schema view, individual vendors might include additional columns or change the order of the columns. Informix recommends that applications not use the forms `SELECT *` or `SELECT table-name*` to access an Information Schema view.

Accessing the Information Schema Views

All Information Schema views have the Select privilege granted to PUBLIC WITH GRANT OPTION so that all users can query the views. Because no other privileges are granted on the Information Schema views, they cannot be updated.

You can query the Information Schema views as you would query any other table or view in the database.

Structure of the Information Schema Views

The following views are described in this section:

- **tables**
- **columns**
- **sql_languages**
- **server_info**

Most of the columns in the views are defined as VARCHAR data types with large maximums to accept large names and in anticipation of long identifier names in future standards.

The tables Information Schema View

The **tables** Information Schema view contains one row for each table to which you have access. It contains the following columns.

Column Name	Data Type	Explanation
table_schema	VARCHAR(128)	Owner of table
table_name	VARCHAR(128)	Name of table or view
table_type	VARCHAR(128)	BASE TABLE for table or VIEW for view
remarks	VARCHAR(255)	Reserved

The visible rows in the **tables** view depend on your privileges. For example, if you have one or more privileges on a table (such as Insert, Delete, Select, References, Alter, Index, or Update on one or more columns), or if these privileges are granted to PUBLIC, you see one row that describes that table.

The columns Information Schema View

The **columns** Information Schema view contains one row for each accessible column. It contains the following columns.

Column Name	Data Type	Explanation
table_schema	VARCHAR(128)	Owner of table
table_name	VARCHAR(128)	Name of table or view
column_name	VARCHAR(128)	Name of the column of the table or view
ordinal_position	INTEGER	Ordinal position of the column. The ordinal_position of a column in a table is a sequential number that starts at 1 for the first column. This column is an Informix extension to XPG4.
data_type	VARCHAR(254)	Data type of the column, such as CHARACTER or DECIMAL

(1 of 2)

Column Name	Data Type	Explanation
char_max_length	INTEGER	Maximum length for character data types; null otherwise
numeric_precision	INTEGER	Total number of digits allowed for exact numeric data types (DECIMAL, INTEGER, MONEY, and SMALLINT), and the number of digits of mantissa precision for approximate data types (FLOAT and SMALLFLOAT), and null for all other data types. The value is machine dependent for FLOAT and SMALLFLOAT.
numeric_prec_radix	INTEGER	Uses one of the following values: Two approximate data types (FLOAT and SMALLFLOAT) 10 exact numeric data types (DECIMAL, INTEGER, MONEY, and SMALLINT) Null for all other data types
numeric_scale	INTEGER	Number of significant digits to the right of the decimal point for DECIMAL and MONEY data types: 0 for INTEGER and SMALLINT data types Null for all other data types
datetime_precision	INTEGER	Number of digits in the fractional part of the seconds for DATE and DATETIME columns; null otherwise. This column is an Informix extension to XPG4.
is_nullable	VARCHAR(3)	Indicates whether a column allows nulls; either YES or NO
remarks	VARCHAR(254)	Reserved

(2 of 2)

The sql_languages Information Schema View

The **sql_languages** Information Schema view contains a row for each instance of conformance to standards that the current database server supports. The **sql_languages** Information Schema view contains the columns that the following table shows.

The **sql_languages** Information Schema view is completely visible to all users.

The server_info Information Schema View

The **server_info** Information Schema view describes the database server to which the application is currently connected. It contains the following columns.

Column Name	Data Type	Explanation
server_attribute	VARCHAR(254)	An attribute of the database server
attribute_value	VARCHAR(254)	Value of the server_attribute as it applies to the current database server

Each row in this view provides information about one attribute. X/Open-compliant databases must provide applications with certain required information about the database server. The **server_info** view includes the information that the following table shows.

server_attribute	Description
identifier_length	Maximum number of characters for a user-defined name
row_length	Maximum length of a row

(1 of 2)

server_attribute	Description
userid_length	Maximum number of characters of a user name (or “authorization identifier”)
txn_isolation	Initial transaction isolation level that the database server assumes: Read Committed Default isolation level for databases created without logging Read Uncommitted Default isolation level for databases created with logging but not ANSI compliant Serializable Default isolation level for ANSI-compliant databases
collation_seq	Assumed ordering of the character set for the database server. The following values are possible: ISO 8859-1 EBCDIC The Informix representation shows ISO 8859-1.

(2 of 2)

The **server_info** Information Schema view is completely visible to all users.

Data Types

Database Data Types	2-3
Summary of Data Types	2-3
BYTE	2-5
CHAR(<i>n</i>)	2-6
CHARACTER(<i>n</i>)	2-8
CHARACTER VARYING(<i>m,r</i>).	2-8
DATE	2-8
DATETIME	2-9
DEC	2-13
DECIMAL	2-13
DOUBLE PRECISION	2-15
FLOAT(<i>n</i>).	2-15
INT	2-15
INTEGER.	2-16
INTERVAL	2-16
MONEY(<i>p,s</i>).	2-19
NCHAR(<i>n</i>)	2-20
NUMERIC(<i>p,s</i>)	2-21
NVARCHAR(<i>m,r</i>)	2-21
REAL	2-21
SERIAL(<i>n</i>)	2-21
SMALLFLOAT	2-22
SMALLINT	2-23
TEXT	2-23
VARCHAR(<i>m,r</i>)	2-25
Data Type Conversions	2-27
Converting from Number to Number	2-27
Converting Between Number and CHAR.	2-28
Converting Between DATE and DATETIME.	2-29

Range of Operations on DATE, DATETIME, and INTERVAL	2-29
Manipulating DATETIME Values.	2-31
Manipulating DATETIME with INTERVAL Values	2-32
Manipulating DATE with DATETIME and INTERVAL Values. . .	2-33
Manipulating INTERVAL Values	2-34
Multiplying or Dividing INTERVAL Values	2-35

Every column in a table is assigned a *data type*. The data type precisely defines the type of values that you can store in that column.

This chapter covers the following topics:

- Database data types
- Data type conversions
- Range of operations on DATE, DATETIME, and INTERVAL

Database Data Types

You assign data types with the CREATE TABLE statement and change them with the ALTER TABLE statement. When you change an existing data type, all data is converted to the new data type, if possible. For more information on the ALTER TABLE and CREATE TABLE statements and data type syntax conventions, refer to the [Informix Guide to SQL: Syntax](#). For a general introduction to data types, see the [Informix Guide to SQL: Tutorial](#).

Summary of Data Types

Informix products recognize the data types that Figure 2-1 lists. The remainder of this chapter describes each of these data types.

Figure 2-1
Data Types That Informix Products Recognize

Data Type	Explanation
BYTE	Stores any kind of binary data
CHAR(<i>n</i>)	Stores single-byte or multibyte sequences of characters, including letters, numbers, and symbols; collation is code-set dependent
CHARACTER(<i>n</i>)	Is a synonym for CHAR
CHARACTER VARYING(<i>m,r</i>)	Stores single-byte and multibyte sequences of characters, including letters, numbers, and symbols of varying length (ANSI compliant); collation is code-set dependent.
DATE	Stores calendar date
DATETIME	Stores calendar date combined with time of day
DEC	Is a synonym for DECIMAL
DECIMAL	Stores numbers with definable scale and precision
DOUBLE PRECISION	Behaves the same way as FLOAT
FLOAT(<i>n</i>)	Stores double-precision floating-point numbers corresponding to the double data type in C
INT	Is a synonym for INTEGER
INTEGER	Stores whole numbers from $-2,147,483,647$ to $+2,147,483,647$
INTERVAL	Stores span of time
MONEY(<i>p,s</i>)	Stores currency amount
NCHAR(<i>n</i>)	Stores single-byte and multibyte sequences of characters, including letters, numbers, and symbols; collation is locale dependent
NUMERIC(<i>p,s</i>)	Is a synonym for DECIMAL
NVARCHAR(<i>m,r</i>)	Stores single-byte and multibyte sequences of characters, including letters, numbers, and symbols of varying length; collation is locale dependent

(1 of 2)

Data Type	Explanation
REAL	Is a synonym for SMALLFLOAT
SERIAL	Stores sequential integers
SMALLFLOAT	Stores single-precision floating-point numbers corresponding to the float data type in C
SMALLINT	Stores whole numbers from $-32,767$ to $+32,767$
TEXT	Stores any kind of text data.
VARCHAR(<i>m,r</i>)	Stores single byte strings of letters, numbers, and symbols of varying length; collation is code-set dependent. Collation is code-set dependent.

(2 of 2)

BYTE

The BYTE data type stores any kind of binary data in an undifferentiated byte stream. Binary data typically consists of saved spreadsheets, program load modules, digitized voice patterns, and so on.

The term simple large object is also used to refer to BYTE as well as TEXT data types.

The BYTE data type has no maximum size. A BYTE column has a theoretical limit of 2^{31} bytes and a practical limit that your disk capacity determines.

You can store, retrieve, update, or delete the contents of a BYTE column. However, you cannot use BYTE data items in arithmetic or string operations, or assign literals to BYTE items with the SET clause of the UPDATE statement. You also cannot use BYTE items in any of the following ways:

- With aggregate functions
- With the IN clause
- With the MATCHES or LIKE clauses
- With the GROUP BY clause
- With the ORDER BY clause

You can use BYTE objects in a Boolean expression only if you are testing for null values.

You can insert data into BYTE columns in the following ways:

- With the **dbload** or **onload** utilities
- With the LOAD statement (DB-Access)
- From BYTE host variables (INFORMIX-ESQL/C)

You cannot use a quoted text string, number, or any other actual value to insert or update BYTE columns.

When you select a BYTE column, you can choose to receive all or part of it. To see it all, use the regular syntax for selecting a column. You can also select any part of a BYTE column by using subscripts, as the following example shows:

```
SELECT cat_picture [1,75] FROM catalog WHERE catalog_num = 10001
```

This statement reads the first 75 bytes of the **cat_picture** column associated with the catalog number 10001.



***Tip:** If you select a BYTE column using the DB-Access Interactive Schema Editor, only the phrase “BYTE value” is returned; no actual value is displayed.*

CHAR(*n*)

The CHAR data type stores any sequence of letters, numbers, and symbols. It can store single-byte and multibyte characters, based on what the chosen locale supports. For more information on multibyte CHARs, see [“Multibyte Characters with CHAR” on page 2-7](#).

A character column has a maximum length *n* bytes, where $1 \leq n \leq 32,767$. If you do not specify *n*, CHAR(1) is assumed. Character columns typically store names, addresses, phone numbers, and so on.

Because the length of this column is fixed, when a character value is retrieved or stored, exactly *n* bytes of data are transferred. If the character string is shorter than *n* bytes, the string is extended with spaces to make up the length. If the string value is longer than *n* bytes, the string is truncated, without the database server raising an exception.

GLS

Collating CHAR Data

The collation order of the CHAR data type depends on the code set. That is, this data is sorted by the order of characters as they appear in the code set. For more information, see the [Informix Guide to GLS Functionality](#).

GLS

Multibyte Characters with CHAR

The database locale must support the multibyte characters that a database uses. If you are storing multibyte characters, make sure to calculate the number of bytes needed. For more information on multibyte characters and locales, see the [Informix Guide to GLS Functionality](#).

Treating CHAR Values as Numeric Values

If you plan to perform calculations on numbers stored in a column, you should assign a number data type to that column. Although you can store numbers in CHAR columns, you might not be able to use them in some arithmetic operations. For example, if you insert the sum of values into a character column, you might experience overflow problems if the character column is too small to hold the value. In this case, the insert fails. However, numbers that have leading zeros (such as some zip codes) have the zeros stripped if they are stored as number types INTEGER or SMALLINT. Instead, store these numbers in CHAR columns.

CHAR values are compared to other CHAR values by taking the shorter value and padding it on the right with spaces until the values have equal length. Then the two values are compared for the full length. Comparisons use the code-set collation order.

Nonprintable Characters with CHAR

A CHAR value can include tabs, spaces, and other nonprintable characters. However, you must use an application to insert the nonprintable characters into host variables and to insert the host variables into your database. After passing nonprintable characters to the database server, you can store or retrieve the characters. When you select nonprintable characters, fetch them into host variables and display them with your own display mechanism.

CHARACTER(*n*)

The only nonprintable character that you can enter and display with DB-Access is a tab. If you try to display other nonprintable characters with DB-Access, your screen returns inconsistent results.

CHARACTER(*n*)

The CHARACTER data type is a synonym for CHAR.

CHARACTER VARYING(*m,r*)

The CHARACTER VARYING data type stores any multibyte string of letters, numbers, and symbols of varying length, where *m* is the maximum size of the column and *r* is the minimum amount of space reserved for that column.

The CHARACTER VARYING data type complies with ANSI standards; the Informix VARCHAR data type supports the same functionality. See the description of the VARCHAR data type on [page 2-25](#).

DATE

The DATE data type stores the calendar date. DATE data types require 4 bytes. A calendar date is stored internally as an integer value equal to the number of days since December 31, 1899.

Because DATE values are stored as integers, you can use them in arithmetic expressions. For example, you can subtract a DATE value from another DATE value. The result, a positive or negative INTEGER value, indicates the number of days that elapsed between the two dates.

The following example shows the default display format of a DATE column:

mm/ dd/ yyyy

In this example, *mm* is the month (1-12), *dd* is the day of the month (1-31), and *yyyy* is the year (0001-9999). For the month, Informix products accept a number value 1 or 01 for January, and so on. For the day, Informix products accept a value 1 or 01 that corresponds to the first day of the month, and so on.

GLS

If you enter only a two-digit value for the year, how Informix products fill in the century digits depends on how you set the **DBCENTURY** environment variable. For example, if you enter the year value as 98, whether that year value is stored as 1998 or 2098 depends on the setting of your **DBCENTURY** variable. If you do not set the **DBCENTURY** environment variable, then your Informix products consider the present century as the default. For information on how to set the **DBCENTURY** environment variable, refer to [page 3-27](#).

If you specify a locale other than the default locale, you can display culture-specific formats for dates. The locales and the **GL_DATE** and **DB_DATE** environment variables affect the display formatting of DATE values. They do not affect the internal format used in a DATE column of a database. To change the default DATE format, set the **DBDATE** or **GL_DATE** environment variable. GLS functionality permits the display of international DATE formats. For more information, see the [Informix Guide to GLS Functionality](#). ♦

DATETIME

The DATETIME data type stores an instant in time expressed as a calendar date and time of day. You choose how precisely a DATETIME value is stored; its precision can range from a year to a fraction of a second.

The DATETIME data type is composed of a contiguous sequence of fields that represents each component of time that you want to record and uses the following syntax:

```
DATETIME largest_qualifier TO smallest_qualifier
```

The *largest_qualifier* and *smallest_qualifier* can be any one of the fields that Figure 2-2 lists.

Figure 2-2
DATETIME Field Qualifiers

Qualifier Field	Valid Entries
YEAR	A year numbered from 1 to 9,999 (A.D.)
MONTH	A month numbered from 1 to 12
DAY	A day numbered from 1 to 31, as appropriate to the month
HOURL	An hour numbered from 0 (midnight) to 23
MINUTE	A minute numbered from 0 to 59
SECOND	A second numbered from 0 to 59
FRACTION	A decimal fraction of a second with up to 5 digits of precision. The default precision is 3 digits (a thousandth of a second). To indicate explicitly other precisions, write FRACTION(<i>n</i>), where <i>n</i> is the desired number of digits from 1 to 5.

A DATETIME column does not need to include all fields from YEAR to FRACTION; it can include a subset of fields or even a single field. For example, you can enter a value of MONTH TO HOUR in a column that is defined as YEAR TO MINUTE, as long as each entered value contains information for a contiguous sequence of fields. You cannot, however, define a column for just MONTH and HOUR; this entry must also include a value for DAY.

If you use the DB-Access TABLE menu, and you do not specify the DATETIME qualifiers, the default DATETIME qualifier, YEAR TO YEAR, is assigned.

A valid DATETIME literal must include the DATETIME keyword, the values to be entered, and the field qualifiers. You must include these qualifiers because, as noted earlier, the value you enter can contain fewer fields than defined for that column. Acceptable qualifiers for the first and last fields are identical to the list of valid DATETIME fields that Figure 2-2 lists.

Write values for the field qualifiers as integers and separate them with delimiters. Figure 2-3 lists the delimiters that are used with DATETIME values in the U.S. ASCII English locale.

Figure 2-3
Delimiters Used with DATETIME

Delimiter	Placement in DATETIME Expression
Hyphen	Between the YEAR, MONTH, and DAY portions of the value
Space	Between the DAY and HOUR portions of the value
Colon	Between the HOUR and MINUTE and the MINUTE and SECOND portions of the value
Decimal point	Between the SECOND and FRACTION portions of the value

Figure 2-4 shows a DATETIME YEAR TO FRACTION(3) value with delimiters.

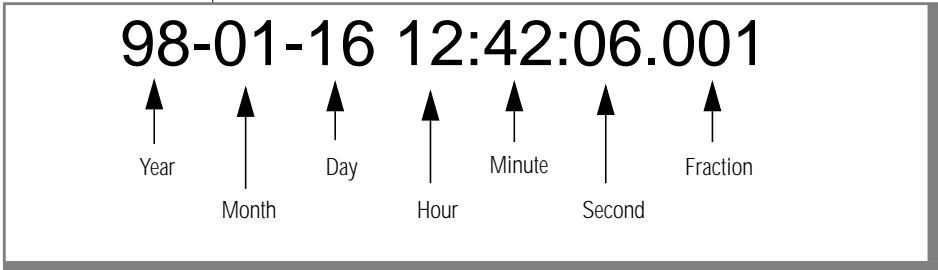


Figure 2-4
Example DATETIME Value with Delimiters

When you enter a value with fewer fields than the defined column, the value that you enter is expanded automatically to fill all the defined fields. If you leave out any more significant fields, that is, fields of larger magnitude than any value that you supply, those fields are filled automatically with the current date. If you leave out any less-significant fields, those fields are filled with zeros (or a 1 for MONTH and DAY) in your entry.

You can also enter DATETIME values as character strings. However, the character string must include information for each field defined in the DATETIME column. THE INSERT statement in the following example shows a DATETIME value entered as a character string:

```
INSERT into cust_calls (customer_num, call_dtime, user_id,
                        call_code, call_descr)
VALUES (101, '1998-01-14 08:45', 'maryj', 'D',
        'Order late - placed 6/1/97')
```

In this example, the **call_dtime** column is defined as DATETIME YEAR TO MINUTE. This character string must include values for the year, month, day, hour, and minute fields. If the character string does not contain information for all defined fields (or adds additional fields), the database server returns an error.

All fields of a DATETIME column are two-digit numbers except for the year and fraction fields. The year field is stored as four digits. When you enter a two-digit value in the year field, how the century digits are filled in and interpreted depends on the value that you assign to the **DBCENTURY** environment variable. For example, if you enter 98 as the year value, whether the year is interpreted as 1998 or 2098 depends on the setting of the **DBCENTURY** variable. If you do not set the **DBCENTURY** environment variable, then your Informix products consider the present century to be the default. For information on how to set and use the **DBCENTURY** environment variable, see [page 3-27](#).

The fraction field requires n digits where $1 \leq n \leq 5$, rounded up to an even number. You can use the following formula (rounded up to a whole number of bytes) to calculate the number of bytes that a DATETIME value requires:

$$\text{total number of digits for all fields} / 2 + 1$$

For example, a YEAR TO DAY qualifier requires a total of eight digits (four for year, two for month, and two for day). This data value requires 5, or $(8/2) + 1$, bytes of storage.

For information on how to use DATETIME data in arithmetic and relational expressions, see [“Range of Operations on DATE, DATETIME, and INTERVAL” on page 2-29](#). For more information on the DATETIME datatype, see the [Informix Guide to SQL: Syntax](#) and the [Informix Guide to GLS Functionality](#).

GLS

If you specify a locale other than U.S. ASCII English, the locale defines the culture-specific display formats for DATETIME values. For more information, see the [Informix Guide to GLS Functionality](#). To change the default display format, change the setting of the **GL_DATETIME** environment variable. With an ESQL API, the **DBTIME** environment variable also affects DATETIME formatting. For more information, see [page 3-44](#). Locales and the **GL_DATE** and **DB_DATE** environment variables affect the display of datetime data. They do not affect the internal format used in a DATETIME column. ♦

DEC

The DEC data type is a synonym for DECIMAL.

DECIMAL

The DECIMAL data type can take two forms: DECIMAL(*p*) floating point and DECIMAL(*p*,*s*) fixed point.

DECIMAL Floating Point

The DECIMAL data type stores decimal floating-point numbers up to a maximum of 32 significant digits, where *p* is the total number of significant digits (the *precision*). Specifying precision is optional. If you do not specify the precision (*p*), DECIMAL is treated as DECIMAL(16), a floating decimal with a precision of 16 places. DECIMAL(*p*) has an absolute value range between 10^{-130} and 10^{124} .

If you use an ANSI-compliant database and specify DECIMAL(*p*), the value defaults to DECIMAL(*p*, 0). For more information about fixed-point decimal values, see the following discussion.

DECIMAL Fixed Point

In fixed-point numbers, DECIMAL(*p*,*s*), the decimal point is fixed at a specific place, regardless of the value of the number. When you specify a column of this type, you write its precision (*p*) as the total number of digits that it can store, from 1 to 32. You write its *scale* (*s*) as the total number of digits that fall to the right of the decimal point. All numbers with an absolute value less than $0.5 * 10^{-s}$ have the value zero. The largest absolute value of a variable of this type that you can store without an error is $10^{p-s} - 10^{-s}$. A DECIMAL data type column typically stores numbers with fractional parts that must be stored and displayed exactly (for example, rates or percentages).

DECIMAL Storage

The database server uses 1 byte of disk storage to store two digits of a decimal number. The database server uses an additional byte to store the exponent and sign. The significant digits to the left of the decimal and the significant digits to the right of the decimal are stored on separate groups of bytes. The way the database server stores decimal numbers is best illustrated with an example. If you specify DECIMAL(6,3), the data type consists of three significant digits to the left of the decimal and three significant digits to the right of the decimal (for instance, 123.456). The three digits to the left of the decimal are stored on 2 bytes (where one of the bytes only holds a single digit) and the three digits to the right of the decimal are stored on another 2 bytes, as Figure 2-5 illustrates. (The exponent byte is not shown.) With the additional byte required for the exponent and sign, this data type requires a total of 5 bytes of storage.

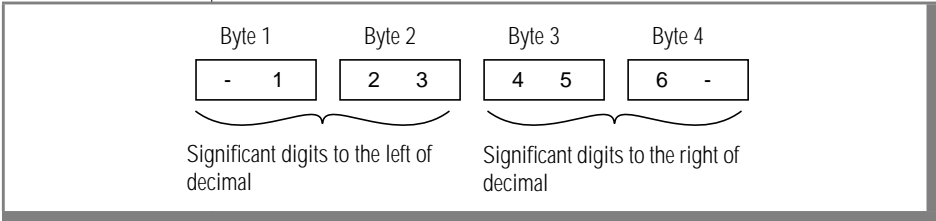


Figure 2-5
Schematic That
Illustrates the
Storage of Digits in
a Decimal Value

You can use the following formulas (rounded *down* to a whole number of bytes) to calculate the byte storage (N) for a decimal data type (N includes the byte required to store the exponent and sign):

If the *scale* is odd: $N = (precision + 4) / 2$
If the *scale* is even: $N = (precision + 3) / 2$

For example, the data type DECIMAL(5,3) requires 4 bytes of storage (9/2 rounded down equals 4).

One caveat to these formulas exists. The maximum number of bytes the database server uses to store a decimal value is 17. One byte is used to store the exponent and sign leaving 16 bytes to store up to 32 digits of precision. If you specify a precision of 32 and an *odd* scale, however, you lose 1 digit of precision. Consider, for example, the data type DECIMAL(32,31). This decimal is defined as 1 digit to the left of the decimal and 31 digits to the right. The 1 digit to the left of the decimal requires 1 byte of storage. This leaves only 15 bytes of storage for the digits to the right of the decimal. The 15 bytes can accommodate only 30 digits, so 1 digit of precision is lost.

DOUBLE PRECISION

Columns defined as DOUBLE PRECISION behave the same as those defined as FLOAT.

FLOAT(*n*)

The FLOAT data type stores double-precision floating-point numbers with up to 16 significant digits. FLOAT corresponds to the **double** data type in C. The range of values for the FLOAT data type is the same as the range of values for the C **double** data type on your computer.

You can use *n* to specify the precision of a FLOAT data type, but SQL ignores the precision. The value *n* must be a whole number between 1 and 14.

A column with the FLOAT data type typically stores scientific numbers that can be calculated only approximately. Because floating-point numbers retain only their most significant digits, the number that you enter in this type of column and the number the database server displays can differ slightly. The difference between the two values depends on how your computer stores floating-point numbers internally. For example, you might enter a value of 1.1000001 into a FLOAT field and, after processing the SQL statement, the database server might display this value as 1.1. This situation occurs when a value has more digits than the floating-point number can store. In this case, the value is stored in its approximate form with the least significant digits treated as zeros.

FLOAT data types usually require 8 bytes per value.

INT

The INT data type is a synonym for INTEGER.

INTEGER

The INTEGER data type stores whole numbers that range from $-2,147,483,647$ to $2,147,483,647$. The maximum negative number, $-2,147,483,648$, is a reserved value and cannot be used. The INTEGER data type is stored as a signed binary integer and is typically used to store counts, quantities, and so on.

Arithmetic operations and sort comparisons are performed more efficiently on integer data than on float or decimal data. However, INTEGER columns can store only a limited range of values. If the data value exceeds the numeric range, the database server does not store the value.

INTEGER data types require 4 bytes per value.

INTERVAL

The INTERVAL data type stores a value that represents a span of time. INTERVAL types are divided into two classes: *year-month intervals* and *day-time intervals*. A year-month interval can represent a span of years and months, and a day-time interval can represent a span of days, hours, minutes, seconds, and fractions of a second.

An INTERVAL value is always composed of one value, or a contiguous sequence of values, that represents a component of time. The following example defines an INTERVAL data type:

```
INTERVAL largest_qualifier(n) TO smallest_qualifier(n)
```

In this example, the *largest_qualifier* and *smallest_qualifier* fields are taken from one of the two INTERVAL classes, as Figure 2-6 shows, and *n* optionally specifies the precision of the largest field (and smallest field if it is a FRACTION).

Figure 2-6
Interval Classes

Interval Class	Qualifier Field	Valid Entry
YEAR-MONTH INTERVAL	YEAR	A number of years
	MONTH	A number of months
DAY-TIME INTERVAL	DAY	A number of days
	HOURL	A number of hours
	MINUTE	A number of minutes
	SECOND	A number of seconds
	FRACTION	A decimal fraction of a second, with up to 5 digits of precision. The default precision is 3 digits (thousandth of a second). To indicate explicitly other precisions, write FRACTION(<i>n</i>), where <i>n</i> is the desired number of digits from 1 to 5.

As with a DATETIME column, you can define an INTERVAL column to include a subset of the fields that you need; however, because the INTERVAL data type represents a span of time that is independent of an actual date, you cannot combine the two INTERVAL classes. For example, because the number of days in a month depends on which month it is, a single INTERVAL data value cannot combine months and days.

A value entered into an INTERVAL column need not include all fields contained in the column. For example, you can enter a value of HOURL TO SECOND into a column defined as DAY TO SECOND. However, a value must always consist of a contiguous sequence of fields. In the previous example, you cannot enter just HOURL and SECOND values; you must also include MINUTE values.

A valid INTERVAL literal contains the INTERVAL keyword, the values to be entered, and the field qualifiers. (See the discussion of the Literal Interval segment in the [Informix Guide to SQL: Syntax](#).) When a value contains only one field, the largest and smallest fields are the same.

When you enter a value in an INTERVAL column, you must specify the largest and smallest fields in the value, just as you do for DATETIME values. In addition, you can use *n* optionally to specify the precision of the first field (and the last field if it is a FRACTION). If the largest and smallest field qualifiers are both FRACTIONS, you can specify only the precision in the last field. Acceptable qualifiers for the largest and smallest fields are identical to the list of INTERVAL fields that Figure 2-6 displays.

If you use the DB-Access TABLE menu, and you do not specify the INTERVAL field qualifiers, the default INTERVAL qualifier, YEAR TO YEAR, is assigned.

The *largest_qualifier* in an INTERVAL value can be up to nine digits (except for FRACTION, which cannot be more than five digits), but if the value that you want to enter is greater than the default number of digits allowed for that field, you must explicitly identify the number of significant digits in the value that you enter. For example, to define an INTERVAL of DAY TO HOUR that can store up to 999 days, you could specify it the following way:

```
INTERVAL DAY(3) TO HOUR
```

INTERVAL values use the same delimiters as DATETIME values. Figure 2-7 shows the delimiters.

Figure 2-7
INTERVAL Delimiters

Delimiter	Placement in DATETIME Expression
Hyphen	Between the YEAR and MONTH portions of the value
Space	Between the DAY and HOUR portions of the value
Colon	Between the HOUR and MINUTE and the MINUTE and SECOND portions of the value
Decimal point	Between the SECOND and FRACTION portions of the value

You can also enter INTERVAL values as character strings. However, the character string must include information for the identical sequence of fields defined for that column. The INSERT statement in the following example shows an INTERVAL value entered as a character string:

```
INSERT INTO manufact (manu_code, manu_name, lead_time)
VALUES ('BRO', 'Ball-Racquet Originals', '160')
```

Because the **lead_time** column is defined as INTERVAL DAY(3) TO DAY, this INTERVAL value requires only one field, the span of days required for lead time. If the character string does not contain information for all fields (or adds additional fields), the database server returns an error. For more information on entering INTERVAL values as character strings, see the [Informix Guide to SQL: Syntax](#).

By default, all fields of an INTERVAL column are two-digit numbers except for the year and fraction fields. The year field is stored as four digits. The fraction field requires *n* digits where $1 \leq n \leq 5$, rounded up to an even number. You can use the following formula (rounded up to a whole number of bytes) to calculate the number of bytes required for an INTERVAL value:

$$\text{total number of digits for all fields} / 2 + 1$$

For example, a YEAR TO MONTH qualifier requires a total of six digits (four for year and two for month). This data value requires 4, or $(6/2) + 1$, bytes of storage.

For information on using INTERVAL data in arithmetic and relational operations, see “[Range of Operations on DATE, DATETIME, and INTERVAL](#)” on page 2-29. For information on using INTERVAL as a constant expression, see the description of the INTERVAL Field Qualifier segment in the [Informix Guide to SQL: Syntax](#).

MONEY(*p,s*)

The MONEY data type stores currency amounts. As with the DECIMAL data type, the MONEY data type stores fixed-point numbers up to a maximum of 32 significant digits, where *p* is the total number of significant digits (the precision) and *s* is the number of digits to the right of the decimal point (the scale).

Unlike the DECIMAL data type, the MONEY data type always is treated as a fixed-point decimal number. The database server defines the data type MONEY(*p*) as DECIMAL(*p*,2). If the precision and scale are not specified, the database server defines a MONEY column as DECIMAL(16,2).

You can use the following formula (rounded up to a whole number of bytes) to calculate the byte storage for a MONEY data type:

If the *scale* is odd: $N = (\text{precision} + 4) / 2$
 If the *scale* is even: $N = (\text{precision} + 3) / 2$

For example, a MONEY data type with a precision of 16 and a scale of 2 (MONEY(16,2)) requires 10, or $(16 + 3)/2$, bytes of storage.

GLS

The default value that the database server uses for scale is locale-dependent. The default locale specifies a default scale of two. For nondefault locales, if the scale is omitted from the declaration, the database server creates MONEY values with a locale-specific scale. For more information, see the [Informix Guide to GLS Functionality](#). ♦

Client applications format values in MONEY columns with the following currency notation:

- A currency symbol: a dollar sign (\$) at the front of the value
- A thousands separator: a comma (,) that separates every three digits of the value
- A decimal point: a period (.)

GLS

The currency notation that client applications use is locale-dependent. If you specify a nondefault locale, the client uses a culture-specific format for MONEY values. For more information, see the [Informix Guide to GLS Functionality](#). ♦

To change the format for MONEY values, change the **DBMONEY** environment variable. For information on how to set the **DBMONEY** environment variable, see [page 3-36](#).

GLS

NCHAR(*n*)

The NCHAR data type stores fixed-length character data. This data can be a sequence of single-byte or multibyte letters, numbers, and symbols. The main difference between CHAR and NCHAR data types is the collation order. While the collation order of the CHAR data type is defined by the code-set order, the collation order of the NCHAR data type depends on the locale-specific localized order. For more information about the NCHAR data type, see the [Informix Guide to GLS Functionality](#).

NUMERIC(*p,s*)

The NUMERIC data type is a synonym for fixed-point DECIMAL.

NVARCHAR(*m,r*)

The NVARCHAR data type stores character data of varying lengths. This data can be a sequence of single-byte or multibyte letters, numbers, and symbols. The main difference between VARCHAR and NVARCHAR data types is the collation order. While the collation order of the VARCHAR data type is defined by the code-set order, the collation order of the NVARCHAR data type depends on the locale-specific localized order. For more information about the NVARCHAR data type, see the [Informix Guide to GLS Functionality](#).

REAL

The REAL data type is a synonym for SMALLFLOAT.

SERIAL(*n*)

The SERIAL data type stores a sequential integer assigned automatically by the database server when a row is inserted. You can define only one SERIAL column in a table. For information on inserting values in SERIAL columns, see the [Informix Guide to SQL: Syntax](#).

The SERIAL data type is not automatically a unique column. You must apply a unique index to this column to prevent duplicate serial numbers.

If you use the interactive schema editor in DB-Access to define the table, a unique index is applied automatically to a SERIAL column.

The default serial starting number is 1, but you can assign an initial value, *n*, when you create or alter the table. You can assign any number greater than 0 as your starting number. The highest serial number that you can assign is 2,147,483,647. If you assign a number greater than 2,147,483,647, you receive a syntax error.

Once a nonzero number is assigned, it cannot be changed. You can, however, insert a value in a SERIAL column (using the INSERT statement) or reset the serial value *n* (using the ALTER TABLE statement), as long as that value does not duplicate any existing values in the table. When you insert a number in a SERIAL column or reset the next value of a SERIAL column, your database server assigns the next number in sequence to the number entered. However, if you reset the next value of a SERIAL column to a value that is less than the values already in that column, the next value is computed with the following formula:

$$\text{maximum existing value in SERIAL column} + 1$$

For example, if you reset the serial value of the **customer_num** column in the **customer** table to 50 and the highest-assigned customer number is 128, the next customer number assigned is 129.

A SERIAL data column is commonly used to store unique numeric codes (for example, order, invoice, or customer numbers). SERIAL data values require 4 bytes of storage.

SMALLFLOAT

The SMALLFLOAT data type stores single-precision floating-point numbers with approximately eight significant digits. SMALLFLOAT corresponds to the **float** data type in C. The range of values for a SMALLFLOAT data type is the same as the range of values for the C **float** data type on your computer.

A SMALLFLOAT data type column typically stores scientific numbers that can be calculated only approximately. Because floating-point numbers retain only their most significant digits, the number that you enter in this type of column and the number the database displays might differ slightly depending on how your computer stores floating-point numbers internally. For example, you might enter a value of 1.1000001 in a SMALLFLOAT field and, after processing the SQL statement, the application development tool might display this value as 1.1. This difference occurs when a value has more digits than the floating-point number can store. In this case, the value is stored in its approximate form with the least significant digits treated as zeros.

SMALLFLOAT data types usually require 4 bytes per value.

SMALLINT

The SMALLINT data type stores small whole numbers that range from –32,767 to 32,767. The maximum negative number, –32,768, is a reserved value and cannot be used. The SMALLINT value is stored as a signed binary integer.

Integer columns typically store counts, quantities, and so on. Because the SMALLINT data type takes up only 2 bytes per value, arithmetic operations are performed efficiently. However, this data type stores a limited range of values. If the values exceed the range between the minimum and maximum numbers, the database server does not store the value and provides you with an error message.

TEXT

The TEXT data type stores any kind of text data. It can contain both single and multibyte characters. For more information on multibyte characters of TEXT data type, see [“Multibyte Characters with TEXT” on page 2-25](#).

The TEXT data type has no maximum size. A TEXT column has a theoretical limit of 2^{31} bytes and a practical limit that your available disk storage determines.

The term simple-large object is also used to refer to TEXT as well as BYTE data types.

TEXT columns typically store memos, manual chapters, business documents, program source files, and so on. In the default locale U.S. ASCII English, data object of type TEXT can contain a combination of printable ASCII characters and the following control characters:

- Tabs (CTRL-I)
- New lines (CTRL-J)
- New pages (CTRL-L)

You can store, retrieve, update, or delete the contents of a TEXT column. However, you cannot use TEXT data items in arithmetic or string operations or assign literals to TEXT items with the SET clause of the UPDATE statement. You also cannot use TEXT items in the following ways:

- With aggregate functions
- With the IN clause
- With the MATCHES or LIKE clauses
- With the GROUP BY clause
- With the ORDER BY clause

You can use TEXT objects in Boolean expressions only if you are testing for null values.

You can insert data in TEXT columns in the following ways:

- With the **dbload** or **onload** utilities
- With the LOAD statement (DB-Access)
- From TEXT host variables (INFORMIX-ESQL/C)

You cannot use a quoted text string, number, or any other actual value to insert or update TEXT columns.

When you select a TEXT column, you can choose to receive all or part of it. To see all of a column, use the regular syntax for selecting a column into a variable. You can also select any part of a TEXT column with subscripts, as the following example shows:

```
SELECT cat_descr [1,75] FROM catalog WHERE catalog_num = 10001
```

This statement reads the first 75 bytes of the **cat_descr** column associated with catalog number 10001.

Nonprintable Characters with TEXT

Both printable and non-printable characters can be inserted in text columns. Informix products do not do any checking of the data that is inserted in a column with the TEXT data type. For detailed information on entering and displaying nonprintable characters, refer to [“Nonprintable Characters with CHAR” on page 2-7](#).

Multibyte Characters with TEXT

The database locale must support multibyte TEXT characters. For more information, see the [Informix Guide to GLS Functionality](#).

Collating TEXT Data

TEXT data type is collated in code-set order. For more information on collation orders, see the [Informix Guide to GLS Functionality](#).

VARCHAR(*m,r*)

The VARCHAR data type stores single-byte and multibyte character sequences of varying length, where *m* is the maximum byte size of the column and *r* is the minimum amount of byte space reserved for that column. For more information on multibyte VARCHAR sequences, see “[Multibyte Characters with VARCHAR](#)” on [page 2-26](#).

The VARCHAR data type is the Informix implementation of a character varying data type.

The ANSI standard data type for varying character strings is CHARACTER VARYING, described on [page 2-8](#).

You must specify the maximum size (*m*) of the VARCHAR column. The size of this parameter can range from 1 to 255 bytes. If you are placing an index on a VARCHAR column, the maximum size is 254 bytes. You can store shorter, but not longer, character strings than the value that you specify.

Specifying the minimum reserved space (*r*) parameter is optional. This value can range from 0 to 255 bytes but must be less than the maximum size (*m*) of the VARCHAR column. If you do not specify a minimum space value, it defaults to 0. You should specify this parameter when you initially intend to insert rows with short or null data in this column, but later expect the data to be updated with longer values.

Although the use of VARCHAR economizes on space used in a table, it has no effect on the size of an index. In an index based on a VARCHAR column, each index key has length *m*, the maximum size of the column.

When you store a VARCHAR value in the database, only its defined characters are stored. The database server does not strip a VARCHAR object of any user-entered trailing blanks, nor does the database server pad the VARCHAR to the full length of the column. However, if you specify a minimum reserved space (*r*) and some data values are shorter than that amount, some space reserved for rows goes unused.

VARCHAR values are compared to other VARCHAR values and to character values in the same way that character values are compared. The shorter value is padded on the right with spaces until the values have equal lengths; then they are compared for the full length.

GLS

Multibyte Characters with VARCHAR

The database locale must support multibyte VARCHAR characters. If you store multibyte characters, make sure to calculate the number of bytes needed. For more information, see the [Informix Guide to GLS Functionality](#).

GLS

Collating VARCHAR

The main difference between the NVARCHAR and the VARCHAR data types is the difference in collation sequencing. Collation order of NVARCHAR characters depends on the GLS locale chosen, while collation of VARCHAR characters depends on the code set. For more information, see the [Informix Guide to GLS Functionality](#).

Nonprintable Characters with VARCHAR

Nonprintable VARCHAR characters are entered, displayed, and treated in the same way as nonprintable CHAR characters are. For detailed information on entering and displaying nonprintable characters, refer to “[Nonprintable Characters with CHAR](#)” on page 2-7.

Storing Numeric Values in a VARCHAR Column

When you insert a numeric value in a VARCHAR column, the stored value does not get padded with trailing blanks to the maximum length of the column. The number of digits in a numeric VARCHAR value is the number of characters that you need to store that value. For example, given the following statement, the value that gets stored in table **mytab** is 1.

```
create table mytab (col1 varchar(10));
insert into mytab values (1);
```

Tip: VARCHAR treats C null (binary 0) and string terminators as termination characters for nonprintable characters.



Data Type Conversions

You might want to change the data type of a column when you need to store larger values than the current data type can accommodate. For example, if you create a SMALLINT column and later find that you need to store integers larger than 32,767, you must change the data type of that column to store the larger value. You can use the ALTER TABLE statement to change the data type of that column.

If you change data types, the new data type must be able to store all the old values. For example, if you try to convert a column from the INTEGER data type to the SMALLINT data type and the following values exist in the INTEGER column, the database server does not change the data type because SMALLINT columns cannot accommodate numbers greater than 32,768:

```
100 400 700 50000700
```

The same situation might occur if you attempt to transfer data from FLOAT or SMALLFLOAT columns to INTEGER, SMALLINT, or DECIMAL columns.

Converting from Number to Number

When you convert columns from one number data type to another, you occasionally find rounding errors. Figure 2-8 indicates which numeric data type conversions are acceptable and what kinds of errors you can encounter when you convert between certain numeric data types.

Figure 2-8
Numeric Data Type Conversion Chart

From:	To:				
	SMALLINT	INTEGER	SMALLFLOAT	FLOAT	DECIMAL
SMALLINT	OK	OK	OK	OK	O
INTEGER	X	OK	X	OK	O
SMALLFLOAT	X	X	OK	OK	O
FLOAT	X	X	F	OK	O
DECIMAL	X	X	F	F	O

Legend:

- OK = No error
- O = An error can occur depending on precision of the decimal
- X = An error can occur depending on data
- F = No error, but less significant digits might be lost

For example, if you convert a FLOAT column to DECIMAL(4,2), your database server rounds off the floating-point numbers before it stores them as decimal numbers. This conversion can result in an error depending on the precision assigned to the DECIMAL column.

Converting Between Number and CHAR

You can convert a CHAR (or NCHAR) column to a number column and vice versa. However, if the CHAR or NCHAR column contains any characters that are not valid in a number column (for example, the letter *l* instead of the number *1*), your database server cannot complete the ALTER TABLE statement and leaves the column values as characters. You receive an error message, and the statement is rolled back, whether you are in a transaction or not.

Converting Between DATE and DATETIME

You can convert DATE columns to DATETIME columns. However, if the DATETIME column contains more fields than the DATE column, the database server either ignores the fields or fills them with zeros. The illustrations in the following list show how these two data types are converted (assuming that the default date format is *mm/dd/yyyy*):

- If you convert DATE to DATETIME YEAR TO DAY, the database server converts the existing DATE values to DATETIME values. For example, the value 01/15/1998 becomes 1998-01-15.
- If you convert DATETIME YEAR TO DAY to the DATE FORMAT, the value 1998-01-15 becomes 01/15/1998.
- If you convert DATE to DATETIME YEAR TO SECOND, the database server converts existing DATE values to DATETIME values and fills in the additional DATETIME fields with zeros. For example, 01/15/1998 becomes 1998-01-15 00:00:00.
- If you convert DATETIME YEAR TO SECOND to DATE, the database server converts existing DATETIME to DATE values but drops fields more precise than DAY. For example, 1998-01-15 12:15:37 becomes 01/15/1998.

Range of Operations on DATE, DATETIME, and INTERVAL

You can use DATE, DATETIME, and INTERVAL data in a variety of arithmetic and relational expressions. You can manipulate a DATETIME value with another DATETIME value, an INTERVAL value, the current time (identified by the keyword CURRENT), or a specified unit of time (identified by the keyword UNITS). In most situations, you can use a DATE value wherever it is appropriate to use a DATETIME value and vice versa. You can also manipulate an INTERVAL value with the same choices as a DATETIME value. In addition, you can multiply or divide an INTERVAL value by a number.

An INTERVAL column can hold a value that represents the difference between two DATETIME values or the difference between (or sum of) two INTERVAL values. In either case, the result is a span of time, which is an INTERVAL value. On the other hand, if you add or subtract an INTERVAL value from a DATETIME value, another DATETIME value is produced because the result is a specific point in time.

Figure 2-9 indicates the range of expressions that you can use with DATE, DATETIME, and INTERVAL data, along with the data type that results from each expression.

Figure 2-9
Range of Expressions for DATE, DATETIME, and INTERVAL

Data Type of Operand 1	Operator	Data Type of Operand 2	Result
DATE	—	DATETIME	INTERVAL
DATETIME	—	DATE	INTERVAL
DATE	+ or —	INTERVAL	DATETIME
DATETIME	—	DATETIME	INTERVAL
DATETIME	+ or —	INTERVAL	DATETIME
INTERVAL	+	DATETIME	DATETIME
INTERVAL	+ or —	INTERVAL	INTERVAL
DATETIME	—	CURRENT	INTERVAL
CURRENT	—	DATETIME	INTERVAL
INTERVAL	+	CURRENT	DATETIME
CURRENT	+ or —	INTERVAL	DATETIME
DATETIME	+ or —	UNITS	DATETIME
INTERVAL	+ or —	UNITS	INTERVAL
INTERVAL	* or /	NUMBER	INTERVAL

No other combinations are allowed. You cannot add two DATETIME values because this operation does not produce either a point in time or a span of time. For example, you cannot add December 25 and January 1, but you can subtract one from the other to find the time span between them.

Manipulating DATETIME Values

You can subtract most DATETIME values from each other. Dates can be in any order and the result is either a positive or a negative INTERVAL value. The first DATETIME value determines the field precision of the result.

If the second DATETIME value has fewer fields than the first, the shorter value is extended automatically to match the longer one. (See the discussion of the EXTEND function in the “Expression” segment in the [Informix Guide to SQL: Syntax](#).) In the following example, subtracting the DATETIME YEAR TO HOUR value from the DATETIME YEAR TO MINUTE value results in a positive interval value of 60 days, 1 hour, and 30 minutes. Because minutes were not included in the second value, the database server sets the minutes for the result to 0.

```
DATETIME (1998-2-15 12:30) YEAR TO MINUTE  
- DATETIME (1998-1-30 11) YEAR TO HOUR
```

```
Result: INTERVAL (15 01:30) DAY TO MINUTE
```

If the second DATETIME value has more fields than the first (regardless of whether the precision of the extra fields is larger or smaller than those in the first value), the additional fields in the second value are ignored in the calculation.

In the following expression (and result), the year is not included for the second value. Therefore, the year is set automatically to the current year, in this case 1998, and the resulting INTERVAL is negative, indicating that the second date is later than the first.

```
DATETIME (1998-9-30) YEAR TO DAY  
- DATETIME (10-1) MONTH TO DAY
```

```
Result: INTERVAL (1) DAY TO DAY [assuming current year  
is 1998]
```

Manipulating DATETIME with INTERVAL Values

INTERVAL values can be added to or subtracted from DATETIME values. In either case, the result is a DATETIME value. If you add an INTERVAL value to a DATETIME value, the order of values is unimportant; however, if you subtract, the DATETIME value must come first. Adding or subtracting an INTERVAL value simply moves the DATETIME value forward or backward in time. The expression that the following example shows moves the date ahead three years and five months:

```
DATETIME (1994-8-1) YEAR TO DAY  
+ INTERVAL (3-5) YEAR TO MONTH  
  
Result: DATETIME (1998-01-01) YEAR TO DAY
```



Important: Evaluate the logic of your addition or subtraction. Remember that months can be 28, 29, 30, or 31 days and that years can be 365 or 366 days.

In most situations, the database server automatically adjusts the calculation when the initial values do not have the same precision. However, in certain situations, you must explicitly adjust the precision of one value to perform the calculation. If the INTERVAL value you are adding or subtracting has fields that are not included in the DATETIME value, you must use the EXTEND function to explicitly extend the field qualifier of the DATETIME value. (For more information on the EXTEND function, see the Expression segment in [Informix Guide to SQL: Syntax](#).) For example, you cannot subtract a minute INTERVAL value from the DATETIME value in the previous example that has a YEAR TO DAY field qualifier. You can, however, use the EXTEND function to perform this calculation, as the following example shows:

```
EXTEND (DATETIME (1998-8-1) YEAR TO DAY, YEAR TO MINUTE)  
- INTERVAL (720) MINUTE(3) TO MINUTE  
  
Result: DATETIME (1998-07-31 12:00) YEAR TO MINUTE
```

The EXTEND function allows you to explicitly increase the DATETIME precision from YEAR TO DAY to YEAR TO MINUTE. This allows the database server to perform the calculation, with the resulting extended precision of YEAR TO MINUTE.

Manipulating DATE with DATETIME and INTERVAL Values

You can use DATE values in arithmetic expressions with DATETIME or INTERVAL values by writing expressions that allow the manipulations that Figure 2-10 shows.

Figure 2-10
Results of Expressions That Manipulate DATE with DATETIME or INTERVAL Values

Expression	Result
DATE - DATETIME	INTERVAL
DATETIME - DATE	INTERVAL
DATE + or - INTERVAL	DATETIME

In the cases that Figure 2-10 show, DATE values are first converted to their corresponding DATETIME equivalents, and then the expression is computed normally.

Although you can interchange DATE and DATETIME values in many situations, you must indicate whether a value is a DATE or a DATETIME data type. A DATE value can come from the following sources:

- A column or program variable of type DATE
- The TODAY keyword
- The DATE() function
- The MDY function
- A DATE literal

A DATETIME value can come from the following sources:

- A column or program variable of type DATETIME
- The CURRENT keyword
- The EXTEND function
- A DATETIME literal

When you represent DATE and DATETIME values as quoted character strings, the fields in the strings must be in proper order. In other words, when a DATE value is expected, the string must be in DATE format, and when a DATETIME value is expected, the string must be in DATETIME format. For example, you can use the string '10/30/1998' as a DATE string but not as a DATETIME string. Instead, you must use '1998-10-30' or '98-10-30' as the DATETIME string.

You can also subtract one DATE value from another DATE value, but the result is a positive or negative INTEGER value rather than an INTERVAL value. If an INTERVAL value is required, you can either convert the INTEGER value to an INTERVAL value or one of the DATE values to a DATETIME value before subtracting.

For example, the following expression uses the DATE() function to convert character string constants to DATE values, calculates their difference, and then uses the UNITS DAY keywords to convert the INTEGER result to an INTERVAL value:

```
(DATE ('5/2/1994') - DATE ('4/6/1955')) UNITS DAY
```

```
Result: INTERVAL (12810) DAY(5) TO DAY
```

If you need YEAR TO MONTH precision, you can use the EXTEND function on the first DATE operand, as the following example shows:

```
EXTEND (DATE ('5/2/1994'), YEAR TO MONTH) - DATE ('4/6/1955')
```

```
Result: INTERVAL (39-01) YEAR TO MONTH
```

The resulting INTERVAL precision is YEAR TO MONTH because the DATETIME value came first. If the DATE value had come first, the resulting INTERVAL precision would have been DAY(5) TO DAY.

Manipulating INTERVAL Values

You can add or subtract INTERVAL values as long as both values are from the same class; that is, both are year-month or both are day-time. In the following example, a SECOND TO FRACTION value is subtracted from a MINUTE TO FRACTION value:

```
INTERVAL (100:30.0005) MINUTE(3) TO FRACTION(4)  
- INTERVAL (120.01) SECOND(3) TO FRACTION
```

```
Result: INTERVAL (98:29.9905) MINUTE TO FRACTION(4)
```

Note the use of numeric qualifiers to alert the database server that the MINUTE and FRACTION in the first value and the SECOND in the second value exceed the default number of digits.

When you add or subtract INTERVAL values, the second value cannot have a field with greater precision than the first. The second INTERVAL, however, can have a field of smaller precision than the first. For example, the second INTERVAL can be HOUR TO SECOND when the first is DAY TO HOUR. The additional fields (in this case MINUTE and SECOND) in the second INTERVAL value are ignored in the calculation.

Multiplying or Dividing INTERVAL Values

You can multiply or divide INTERVAL values by a number that can be an integer or a fraction. However, any remainder from the calculation is ignored and the result is truncated. The following expression multiplies an INTERVAL by a fraction:

```
INTERVAL (15:30.0002) MINUTE TO FRACTION(4) * 2.5
```

```
Result: INTERVAL (38:45.0005) MINUTE TO FRACTION(4)
```

In this example, $15 * 2.5 = 37.5$ minutes, $30 * 2.5 = 75$ seconds, and $2 * 2.5 = 5$ fraction(4). The 0.5 minute is converted to 30 seconds and 60 seconds are converted to 1 minute, which produces the final result of 38 minutes, 45 seconds, and 0.0005 of a second. The results of any calculation include the same amount of precision as the original INTERVAL value.

Environment Variables

Types of Environment Variables	3-5
Where to Set Environment Variables in UNIX.	3-6
Environment Variables at the System Prompt	3-6
Environment Variables in an Environment-Configuration File	3-6
Environment Variables at Login Time	3-6
Manipulating Environment Variables in UNIX	3-7
Setting Environment Variables in an Environment-Configuration File	3-7
Setting Environment Variables at Login Time in UNIX	3-8
Syntax for Setting Environment Variables in UNIX	3-8
Unsetting Environment Variables in UNIX	3-9
Modifying the Setting of an Environment Variable in UNIX	3-9
Viewing Your Environment Variable Settings in UNIX	3-10
Checking Environment Variables in UNIX with the chkenv Utility	3-11
Rules of Precedence in UNIX	3-12
Where to Set Environment Variables in Windows NT	3-12
Manipulating Environment Variables in Windows NT	3-13
Setting Environment Variables for Native Windows Applications	3-13
Setting Environment Variables for Command-Prompt Utilities in Windows NT	3-14
Using the System Applet to Work with Environment Variables	3-15

Using the Command Prompt to Work with Environment Variables	3-15
Using dbservername.cmd to Initialize a Command-Prompt Environment in Windows NT	3-17
Rules of Precedence for Environment Variables in Windows NT	3-17
List of Environment Variables	3-18
Environment Variables.	3-22
ARC_DEFAULT	3-22
ARC_KEYPAD	3-23
CPFIRST	3-24
DBANSIWARN	3-25
DBBLOBBUF	3-26
DBCENTURY	3-26
DBDATE	3-29
DBDELIMITER	3-32
DBEDIT	3-32
DBFLTMASK	3-33
DBLANG	3-33
DBMONEY	3-35
DBONPLOAD	3-36
DBPATH	3-37
DBPRINT	3-39
DBREMOTECD	3-40
DBSPACETEMP	3-41
DBTIME	3-43
DBUPSPACE	3-46
DELIMIDENT	3-46
ENVIGNORE	3-47
FET_BUF_SIZE	3-48
IFX_DIRECTIVES	3-48
INFORMIXC	3-50
INFORMIXCONRETRY	3-50
INFORMIXCONTIME	3-51
INFORMIXDIR	3-52
INFORMIXKEYTAB	3-53
INFORMIXOPCACHE	3-54
INFORMIXSERVER	3-54
INFORMIXSHMBASE	3-55
INFORMIXSQLHOSTS	3-56

INFORMIXSTACKSIZE	3-57
INFORMIXTERM	3-58
INF_ROLE_SEP	3-59
NODEFDAC	3-59
ONCONFIG	3-60
OPTCOMPIND	3-61
PATH.	3-62
PDQPRIORITY	3-62
PLCONFIG.	3-64
PSORT_DBTEMP.	3-64
PSORT_NPROCS.	3-65
SQLEXEC	3-67
SQLRM	3-68
SQLRMDIR.	3-68
TERM.	3-69
TERMCAP	3-69
TERMINFO.	3-70
THREADLIB	3-71
Index of Environment Variables	3-71

Various *environment variables* affect the functionality of your Informix products. You can set environment variables that identify your terminal, specify the location of your software, and define other parameters. The environment variables discussed in this chapter are grouped and listed alphabetically beginning on [page 3-18](#). In addition, an index of environment variables is included at the end of this chapter on [page 3-71](#).

Some environment variables are required, and others are optional. For example, you must either set or accept the default setting for certain UNIX environment variables.

This chapter describes how to use the environment variables that apply to one or more Informix products and shows how to set them.

Types of Environment Variables

The environment variables discussed in this chapter include the following categories:

- **Informix-specific environment variables**
Set these standard environment variables when you want to work with Informix products. Each product manual specifies the environment variables that you must set to use that product.
- **Operating-system-specific environment variables**
Informix products rely on the correct setting of certain standard operating-system environment variables. The **PATH** environment variable, for example, must always be set.

In a UNIX environment, you might also have to set the **TERMCAP** or **TERMINFO** environment variable to use some products effectively.

GLS

The GLS environment variables that allow you to work in a nondefault locale are described in the [Informix Guide to GLS Functionality](#). However, these variables are included in the list of environment variables [on page 3-18](#) and in the index table in [Figure 3-1 on page 3-71](#). ♦

UNIX

Where to Set Environment Variables in UNIX

You can set environment variables in the following places:

- At the system prompt on the command line
- In an environment-configuration file
- In a login file

Environment Variables at the System Prompt

When you set an environment variable at the system prompt, you must reassign it the next time you log into the system. For more information, see [“Manipulating Environment Variables in UNIX” on page 3-7](#).

Environment Variables in an Environment-Configuration File

The environment-configuration file is a common or private file where you can define all the environment variables that Informix products use. An environment-configuration file reduces the number of environment variables that you must set at the command line or in a shell file.

Environment Variables at Login Time

When you set an environment variable in your **.login**, **.cshrc**, or **.profile** file, it is assigned automatically every time you log into the system.

UNIX

Manipulating Environment Variables in UNIX

The following sections discuss setting, unsetting, viewing, and modifying environment variables. If you already use an Informix product, some or all of the appropriate environment variables might be set.

Setting Environment Variables in an Environment-Configuration File

The common (shared) environment-configuration file resides in the `$INFORMIXDIR/etc/informix.rc` file. The permission for this shared file must be set to 644. A user can override the system or common environment variables by setting variables in a *private environment-configuration file*. The private environment-configuration file must have the following characteristics:

- The file is stored in the user's home directory
- The file is named **.informix**
- Permissions are set, by the user, to readable

An environment-configuration file can contain comment lines (preceded by `#`) and variable lines and their values (separated by blanks and tabs), as the following example shows:

```
# This is an example of an environment-configuration file
#
DBDATE DMY4-
#
# These are ESQL/C environment variable settings
#
INFORMIXC gcc
CPFIRST TRUE
```

You can use the **ENVIGNORE** environment variable to override one or more entries in this file. Use the Informix **chkenv** utility to perform a sanity check on the contents of an environment-configuration file. The **chkenv** utility returns an error message if the file contains a bad environment-variable entry or if the file is too large. The **chkenv** utility is described on [page 3-11](#).

The first time that you set an environment variable in a shell file or configuration file before you work with your Informix product, you should *source* the file (if you use a C shell) or use a period (.) to execute an environment-configuration file (if you use a Bourne or Korn shell). This procedure tells the shell process to read your entry.

Setting Environment Variables at Login Time in UNIX

Add the commands that set your environment variables to the following login file:

For the C shell **.login** or **.cshrc**
For the Bourne shell or Korn shell **.profile**

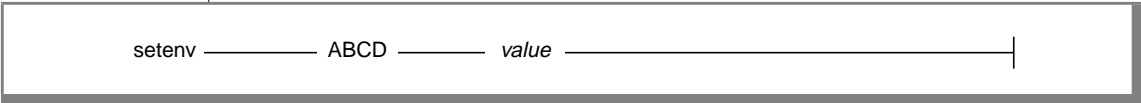
Syntax for Setting Environment Variables in UNIX

Use standard UNIX commands to set environment variables. Depending on the type of shell you use, the following diagram shows how to set the **ABCD** environment variable to *value*. The environment variables are case sensitive.

Shell	Command
C	setenv ABCD value
Bourne	ABCD=value export ABCD
Korn	ABCD=value export ABCD
Korn	export ABCD=value

Korn-shell syntax supports a shortcut, as the last line of the preceding diagram shows.

The following diagram shows how the syntax for setting an environment variable is represented throughout this chapter. These diagrams indicate the setting for the C shell; for the Bourne or Korn shells, use the syntax that the preceding diagram shows.



For more information on how to read syntax diagrams, see “[Command-Line Conventions](#)” in the Introduction.

Unsetting Environment Variables in UNIX

To unset an environment variable, enter the following command.

Shell	Command
C	<code>unsetenv ABCD</code>
Bourne or Korn	<code>unset ABCD</code>

Modifying the Setting of an Environment Variable in UNIX

Sometimes you must add information to an environment variable that is already set. For example, the **PATH** environment variable is always set in UNIX environments. When you use an Informix product, you must add to the **PATH** the name of the directory where the executable files for the Informix products are stored.

In the following example, the **INFORMIXDIR** is **/usr/informix**. (That is, during installation, the Informix products were installed in the **/usr/informix** directory.) The executable files are in the **bin** subdirectory, **/usr/informix/bin**. To add this directory to the front of the C shell **PATH** environment variable, use the following command:

```
setenv PATH /usr/informix/bin:$PATH
```

Rather than entering an explicit pathname, you can use the value of the **INFORMIXDIR** environment variable (represented as **\$INFORMIXDIR**), as the following example shows:

```
setenv INFORMIXDIR /usr/informix
setenv PATH $INFORMIXDIR/bin:$PATH
```

You might prefer to use this version to ensure that your **PATH** entry does not contradict the path that was set in **INFORMIXDIR**, and so that you do not have to reset **PATH** whenever you change **INFORMIXDIR**.

If you set the **PATH** environment variable on the C shell command line, you might need to include curly braces with the existing **INFORMIXDIR** and **PATH**, as the following command shows:

```
setenv PATH ${INFORMIXDIR}/bin:${PATH}
```

For more information about setting and modifying environment variables, refer to the manuals for your operating system.

Viewing Your Environment Variable Settings in UNIX

After one or more Informix products have been installed, enter the following command at the system prompt to view your current environment settings.

UNIX Version	Command
BSD UNIX	env
UNIX System V	printenv

UNIX

Checking Environment Variables in UNIX with the *chkenv* Utility

The **chkenv** utility checks the validity of shared or private environment-configuration files. Use it to provide debugging information when you define, in an environment-configuration file, all the environment variables that your Informix products use.

```
chkenv _____ filename _____
```

Element	Purpose	Key Considerations
<i>filename</i>	Specifies the name of the environment-configuration file that you want to debug.	None

The common environment-configuration file is stored in **\$INFORMIXDIR/etc/informix.rc**. A private environment-configuration file is stored in the user’s home directory as **.informix**.

Issue the following command to check the contents of the shared environment-configuration file:

```
chkenv informix.rc
```

The **chkenv** utility returns an error message if it finds a bad environment-variable entry in the file or if the file is too large. You can modify the file and rerun the utility to check the modified environment-variable settings.

Informix products ignore all lines in the environment-configuration file, starting at the point of the error, if the **chkenv** utility returns the following message:

```
-33500 filename: Bad environment variable on line number.
```

If you want the product to ignore specified environment-variable settings in the file, you can also set the **ENVIGNORE** environment variable. For a discussion of the use and format of environment-configuration files and the **ENVIGNORE** environment variable, see [page 3-48](#).

UNIX

Rules of Precedence in UNIX

When an Informix product accesses an environment variable, normally the following rules of precedence apply:

1. The highest precedence goes to the value that is defined in the environment (shell) by explicitly setting the value at the shell prompt.
2. The second highest precedence goes to the value that is defined in the private environment-configuration file in the user's home directory (`~/.informix`).
3. The next highest precedence goes to the value that is defined in the common environment-configuration file (`$INFORMIXDIR/etc/informix.rc`).
4. The next highest precedence goes to the value that is defined in your `.login` file.
5. The lowest precedence goes to the default value.

For precedence information about GLS environment variables, see the [Informix Guide to GLS Functionality](#).

WIN NT

Where to Set Environment Variables in Windows NT

You might be able to set environment variables in several places in a Windows environment, depending on which Informix application you use.

For native Windows Informix applications, such as the database server, environment variables can be set only in the Windows registry. Environment variables set in the registry cannot be modified elsewhere.

For utilities that run in a command prompt session, such as `dbaccess`, environment variables can be set in several ways, as described in [“Setting Environment Variables for Command-Prompt Utilities in Windows NT”](#) on page 3-14.

To use client applications such as ESQL/C, Relational Object Manager, and NewEra in a Windows environment, use the Setnet32 utility to set environment variables. For information about the Setnet32 utility, see the *Informix Client Products Installation Guide* for your operating system.

Manipulating Environment Variables in Windows NT

The following sections discuss setting, viewing, unsetting, and modifying environment variables for native Windows applications and command-line utilities.

Setting Environment Variables for Native Windows Applications

Native Windows Informix applications, such as the database server itself, store their configuration information in the Windows registry. To modify this information, you must use the Registry Editor, **regedt32.exe**.

Manipulating environment variables with the Registry Editor

1. Launch the Registry Editor, **regedt32.exe**, and choose the window titled **HKEY_LOCAL_MACHINE**.
2. In the left pane, double-click the SOFTWARE registry key (shown as a small, yellow file folder icon). The SOFTWARE registry key expands to show several subkeys, one of which is Informix. Continue down the tree in the following sequence:

`OnLine, dbservername, Environment.`

Substitute the name of your database server in place of *dbservername*.

3. With the Environment registry key selected in the left pane, you should see a list of environment variables and their defined values in the right pane (for example, **CLIENT_LOCALE:REG_SZ:EN_US.CP1252**).

4. Change existing environment variables if needed.
 - a. Double-click the environment variable.
 - b. Type the new value in the String Editor dialog box.
 - c. Click **OK** to accept the value.
5. Add new environment variables if needed.
 - a. Choose **Edit→Add Value** in the Registry Editor.
 - b. Enter the name of the environment variable in the Value Name edit box and choose **REG_SZ** as the data type.
 - c. Click **OK** and type a value for the environment variable in the String Editor dialog box.
6. Delete an environment variable, if needed.
 - a. Select the variable name.
 - b. Choose **Edit→Delete** in the Registry Editor.

For more information on using the Registry Editor, see your operating-system documentation.



Important: *In order to use the Registry Editor to change database server environment variables, you must belong to either the Administrators' or Informix-Admin groups. For information on assigning users to groups, see your operating-system documentation.*

Setting Environment Variables for Command-Prompt Utilities in Windows NT

You can set environment variables for command-prompt utilities in the following ways:

- With the System applet in the Control Panel
- In a command-prompt session

Using the System Applet to Work with Environment Variables

The System applet provides a graphical interface to creating, modifying, and deleting system-wide and user-specific variables. Environment variables that are set with the System applet are visible to all command prompt sessions.

To change environment variables with the System applet in the control panel

1. Double-click the System applet icon from the Control Panel window.
Click the Environment tab near the top of the window. Two list boxes display System Environment Variables and User Environment Variables. System Environment Variables apply to an entire system, and User Environment Variables apply only to the sessions of individual user.
2. To change the value of an existing variable, select that variable.
The name of the variable and its current value appear in the boxes at the bottom of the window.
3. Highlight the existing value and type the new value.
4. To add a new variable, highlight an existing variable and type the new variable name in the box at the bottom of the window.
5. Next, enter the value for the new variable at the bottom of the window and click the **Set** button.
6. To delete a variable, select the variable and click the **Delete** button.



Important: In order to use the System applet to change System environment variables, you must belong to the Administrators' group. For information on assigning users to groups, see your operating-system documentation.

Using the Command Prompt to Work with Environment Variables

The following diagram shows the syntax for setting an environment variable at a command prompt in Windows NT.

_____ s = _____ v |

For more information on how to read syntax diagrams, see “[Command-Line Conventions](#)” in the Introduction.

To view your current settings after one or more Informix products are installed, enter the following command at a command prompt.

_____ s = _____ A _____

Sometimes you must add information to an environment variable that is already set. For example, the **PATH** environment variable is always set in Windows NT environments. When you use an Informix product, you must add the name of the directory where the executable files for the Informix products are stored to the **PATH**.

In the following example, **INFORMIXDIR** is **d:\informix**, (that is, during installation, Informix products were installed in the **d:\informix** directory). The executable files are in the **bin** subdirectory, **d:\informix\bin**. To add this directory at the beginning of the **PATH** environment-variable value, use the following command:

```
set PATH=d:\informix\bin;%PATH%
```

Rather than entering an explicit pathname, you can use the value of the **INFORMIXDIR** environment variable (represented as **%INFORMIXDIR%**), as the following example shows:

```
set INFORMIXDIR=d:\informix
set PATH=%INFORMIXDIR%\bin;%PATH%
```

You might prefer to use this version to ensure that your **PATH** entry does not contradict the path that was set in **INFORMIXDIR** and to avoid resetting **PATH** whenever you change **INFORMIXDIR**.

For more information about setting and modifying environment variables, refer to your operating-system manuals.

Using dbservername.cmd to Initialize a Command-Prompt Environment in Windows NT

Each time that you open a Windows NT command prompt, it acts as an independent environment. Therefore, environment variables that you set within it are valid only for that particular command-prompt instance. For example, if you open one command prompt and set the variable, **INFORMIXDIR**, and then open another command prompt and type `set` to check your environment, you will find that **INFORMIXDIR** is not set in the new command-prompt session.

The database server installation program creates a *batch file* that you can use to configure command-prompt utilities, ensuring that your command-prompt environment is initialized correctly each time that you run a command-prompt session. The batch file, **dbseservername.cmd**, is located in **%INFORMIXDIR%**, and is a plain text file that you can modify with any text editor. If you have more than one database server installed in **%INFORMIXDIR%**, there will be more than one batch file with the **.cmd** extension, each bearing the name of the database server with which it is associated.

To run **dbservername.cmd** from a command prompt, type **dbservername** or configure a command prompt so that it runs **dbservername.cmd** automatically at start up.

Rules of Precedence for Environment Variables in Windows NT

When an Informix product accesses an environment variable, normally the following rules of precedence apply:

1. The highest precedence goes to the value that is defined in the environment by explicitly setting the value at the command prompt.
2. The second highest precedence goes to the value that is defined in the System control panel as a User Environment Variable.
3. The third highest precedence goes to the value that is defined in the System control panel as a System Environment Variable.
4. The lowest precedence goes to the default value.



Important: Because Windows NT services access only environment variables that are set in the registry, the preceding rules of precedence do not apply for Informix native Windows applications. For native Windows applications, the highest precedence goes to variables that are explicitly defined in the registry, and the lowest precedence goes to the default value.

List of Environment Variables

The following table contains an alphabetical list of the environment variables that you can set for an Informix database server and SQL API products. Most of these environment variables are described in this chapter on the pages listed in the last column.

The GLS environment variables that allow you to work in a nondefault locale are described in the [Informix Guide to GLS Functionality](#). However, these variables are included in the list of environment variables on page 3-18 and in the index table in [Figure 3-1 on page 3-71](#).

For precedence information about GLS environment variables, see the [Informix Guide to GLS Functionality](#). ♦

Environment Variable	Restrictions	Page
ARC_DEFAULT	Not for Dynamic Server, Workgroup and Developer Editions Not for Dynamic Server with AD and XP Options UNIX only	3-22
ARC_KEYPAD	Not for Dynamic Server, Workgroup and Developer Editions Not for Dynamic Server with AD and XP Options UNIX only	3-22
CC8BITLEVEL	ESQL/C only	Informix Guide to GLS Functionality

(1 of 5)

Environment Variable	Restrictions	Page
CLIENT_LOCALE	None	Informix Guide to GLS Functionality
CPFIRST	None	3-23
DBANSIWARN	None	3-25
DBBLOBBUF	None	3-26
DBCENTURY	SQL APIs only	3-26
DBDATE	None	3-29; Informix Guide to GLS Functionality
DBDELIMITER	None	3-32
DBEDIT	None	3-32
DBFLTMASK	DB-Access only	3-33
DBLANG	None	3-33; Informix Guide to GLS Functionality
DBMONEY	None	3-35; Informix Guide to GLS Functionality
DBONPLOAD	High-Performance Loader only	3-36
DBPATH	None	3-37
DBPRINT	UNIX only	3-39
DBREMOTECMD	Not for Dynamic Server, Workgroup and Developer Editions UNIX only	3-40
DBSPACETEMP	None	3-41

(2 of 5)

Environment Variable	Restrictions	Page
DBTIME	SQL APIs only	3-43; Informix Guide to GLS Functionality
DBUPSPACE	None	3-46
DB_LOCALE	None	Informix Guide to GLS Functionality
DELIMIDENT	None	3-46
ENVIGNORE	UNIX only	3-48
ESQLMF	None	Informix Guide to GLS Functionality
FET_BUF_SIZE	SQL APIs and DB-Access only	3-48
GLS8BITSYS	None	Informix Guide to GLS Functionality
GL_DATE	None	Informix Guide to GLS Functionality
GL_DATETIME	None	Informix Guide to GLS Functionality
IFX_DIRECTIVES	Informix Dynamic Server only	3-49
INFORMIXC	ESQL/C only UNIX only	3-50
INFORMIXCONRETRY	None	3-50
INFORMIXCONTIME	None	3-51
INFORMIXDIR	None	3-52
INFORMIXKEYTAB	UNIX only	3-53

(3 of 5)

Environment Variable	Restrictions	Page
INFORMIXOPCACHE	Optical Subsystem only	3-54
INFORMIXSERVER	None	3-54
INFORMIXSHMBASE	UNIX only	3-55
INFORMIXSQLHOSTS	None	3-56
INFORMIXSTACKSIZE	None	3-57
INFORMIXTERM	DB-Access only UNIX only	3-58
INF_ROLE_SEP	None	3-59
NODEFDAC	None	3-59
ONCONFIG	None	3-60
OPTCOMPIND	None	3-61
PATH	None	3-62
PDQPRIORITY	Not for Dynamic Server, Workgroup and Developer Editions	3-62
PLCONFIG	Loader	3-64
PSORT_DBTEMP	None	3-64
PSORT_NPROCS	None	3-65
SERVER_LOCALE	None	Informix Guide to GLS Functionality
SQLEXEC	Not for Dynamic Server, Workgroup and Developer Editions or for Dynamic Server with AD and XP options. UNIX only	3-67
SQLRM	(obsolete)	3-68
SQLRMDIR	(obsolete)	3-68

(4 of 5)

List of Environment Variables

Environment Variable	Restrictions	Page
TERM	UNIX only	3-69
TERMCAP	UNIX only	3-69
TERMINFO	UNIX only	3-70
THREADLIB	ESQL/C only UNIX only	3-71

(5 of 5)

Environment Variables

The following sections discuss the environment variables that Informix products use.



IDS

UNIX

W/D

Important: Each description of the environment variables given below includes the syntax for setting the environment variable in the UNIX environment. For a general description of how these environment variables are to be set in Windows NT environments, see [“Setting Environment Variables for Native Windows Applications” on page 3-13](#) and [“Setting Environment Variables for Command-Prompt Utilities in Windows NT” on page 3-14](#).

ARC_DEFAULT

When you use the ON-Archive archive and tape-management system for your database server, you can set the **ARC_DEFAULT** environment variable to indicate where a personal default qualifier file is located.

The **ARC_DEFAULT** environment variable is not available for the Informix Dynamic Server, Workgroup and Developer Editions. ♦

```
setenv _____ ARC_DEFAULT _____ pathname _____
```

pathname is the full pathname of the personal default qualifier file.

For example, to set the **ARC_DEFAULT** environment variable to specify the file **/usr/jane/arcdefault.janeroe**, enter the following command:

```
setenv ARC_DEFAULT /usr/jane/arcdefault.janeroe
```

For more information on archiving, see your [Archive and Backup Guide](#).

IDS

UNIX

W/D

ARC_KEYPAD

If you use the ON-Archive archive and tape-management system for your database server, you can set your **ARC_KEYPAD** environment variable to point to a **ttermcap** file that is different from the default **ttermcap** file. The default is the **\$INFORMIXDIR/etc/ttermcap** file, and it contains instructions on how to modify the **ttermcap** file.

The **ARC_KEYPAD** environment variable is not available for the Informix Dynamic Server, Workgroup and Developer Editions. ♦

The **ttermcap** file serves the following purposes for the ON-Archive menu interface:

- It defines the terminal control attributes that allow ON-Archive to manipulate the screen and cursor.
- It defines the mappings between commands and key presses.
- It defines the characters used in drawing menus and borders for an API.

```
setenv _____ ARC_KEYPAD _____ pathname _____|
```

pathname is the pathname for a **ttermcap** file.

For example, to set the **ARC_KEYPAD** environment variable to specify the file **/usr/jane/ttermcap.janeroe**, enter the following command:

```
setenv ARC_KEYPAD /usr/jane/ttermcap.janeroe
```

For more information on archiving, see your [Archive and Backup Guide](#).

CPFIRST

When compiling an ESQL/C program, the default order is to run the ESQL/C preprocessor on the program source file and then to pass the resulting file to the C language preprocessor and compiler. However, it is also possible to do the compilation of an ESQL/C program source file in the following order:

1. Run the C preprocessor
2. Run the ESQL/C preprocessor
3. Run the C compiler and linker

You can determine the non-default compilation order for a specific program by either giving the program source file a **.ecp** extension or by running the **-cp** option with the **esql** command on a program source file with a **.ec** extension.

It is also possible to determine the non-default compilation order for all ESQL/C source files in your programming environment by setting the **CPFIRST** environment variable. Set the **CPFIRST** environment variable to **TRUE** (uppercase only) to run the C preprocessor on all ESQL/C source files. Then the C preprocessor will run before the ESQL/C preprocessor on all ESQL/C source files in your environment, irrespective of whether the **-cp** option is passed to the **esql** command, or whether the source files have the **.ec** or the **.ecp** extension.

```
setenv _____ CPGFIRST _____ TRUE _____
```

DBANSIWARN

Setting the **DBANSIWARN** environment variable indicates that you want to check for Informix extensions to ANSI standard syntax. Unlike most environment variables, you do not need to set **DBANSIWARN** to a value. You can set it to any value or to no value.

```
setenv _____ DBANSIWARN _____
```

If you set the **DBANSIWARN** environment variable for DB-Access, it is functionally equivalent to including the **-ansi** flag when you invoke the utility from the command line. If you set **DBANSIWARN** before you run DB-Access, warnings are displayed on the screen within the SQL menu.

Set the **DBANSIWARN** environment variable before you compile an INFORMIX-ESQL/C program to check for Informix extensions to ANSI standard syntax. When Informix extensions to ANSI standard syntax are encountered in your program at compile time, warning messages are written to the screen.

At run time, the **DBANSIWARN** environment variable causes the SQL Communication Area (SQLCA) variable **sqlca.sqlwarn.sqlwarn5** to be set to **W** when a statement that is not ANSI-compliant is executed. (For more information on SQLCA, see [INFORMIX-ESQL/C Programmer's Manual](#).)

Once you set **DBANSIWARN**, Informix extension checking is automatic until you log out or unset **DBANSIWARN**. To turn off Informix extension checking, unset the **DBANSIWARN** environment variable by entering the following command:

```
unsetenv DBANSIWARN
```

DBBLOBBUF

The **DBBLOBBUF** environment variable controls whether TEXT or BYTE data is stored temporarily in memory or in a file while being unloaded with the UNLOAD statement.



n represents the maximum size of TEXT or BYTE data in kilobytes.

If TEXT or BYTE data is smaller than the default of 10 kilobytes or the setting of the **DBBLOBBUF** environment variable, it is temporarily stored in memory. If the TEXT or BYTE data is larger than the default or the setting of the environment variable, it is written to a temporary file. This environment variable applies to the UNLOAD command only.

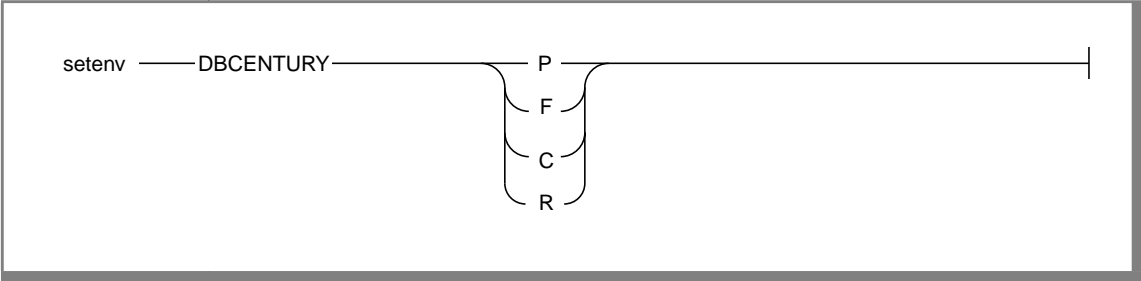
For instance, to set a buffer size of 15 kilobytes, set the **DBBLOBBUF** environment variable as the following example shows:

```
setenv DBBLOBBUF 15
```

In the example, any TEXT or BYTE data that is smaller than 15 kilobytes is stored temporarily in memory. TEXT or BYTE data larger than 15 kilobytes is stored temporarily in a file.

DBCENTURY

The environment variable **DBCENTURY** allows you to choose the appropriate expansion for two-digit year DATE and DATETIME values.



Previously, if only the decade was provided for a literal DATE or DATETIME value in a table column, the present century was used to expand the year. For example, 12/31/97 would expand to 12/31/1997. Three new algorithms now complete the century value of a year: past (P), future (F), and closest (C).

Algorithm	Explanation
P = Past	The past and present centuries are used to expand the year value. These two dates are compared against the current date, and the date that is prior to the current date is chosen. If both dates are prior to the current date, the date that is closest to the current date is chosen.
F = Future	The present and the next centuries are used to expand the year value. These two dates are compared against the current date, and the date that is after the current date is chosen. If both the expansions are after the current date, the date that is closest to the current date is chosen.
C = Closest	The past, present, and next centuries are used to expand the year value, and the date that is closest to the current date is used.
R = Present	The present century is used to expand the year value.

When the **DBCENTURY** environment variable is not set, the current century is used as the system default. To override the default, specify all four digits.

The following examples illustrate how the **DBCENTURY** environment variable expands DATE and DATETIME year formats.

Behavior of DBCENTURY = P

```

Example data type: DATE
Current date: 4/6/1998
User enters: 1-1-1
DBCENTURY = P, Past century algorithm
Previous century expansion : 1/1/1801
Present century expansion: 1/1/1901
Analysis: Both results are prior to the current date, but 1/1/1901 is closer to
the current date. 1/1/1901 is chosen.

```

Behavior of DBCENTURY = F

Example data type: DATETIME year to month
Current date: 5/7/2005
User enters: 1/1/1
DBCENTURY = F, Future century algorithm
Present century expansion: 2001-1
Next century expansion: 2101-1
Analysis: Only date 2101-1 is after the current date and it is chosen as the expansion of the year value.

Behavior of DBCENTURY = C

Example data type: DATE
Current date: 4/6/1998
User enters: 1-1-1
DBCENTURY = C, Closest century algorithm
Previous century expansion : 1/1/1801
Present century expansion: 1/1/1901
Next century expansion: 1/1/2001
Analysis: Because the next century expansion is the closest to the current date, 1/1/2001 is chosen.

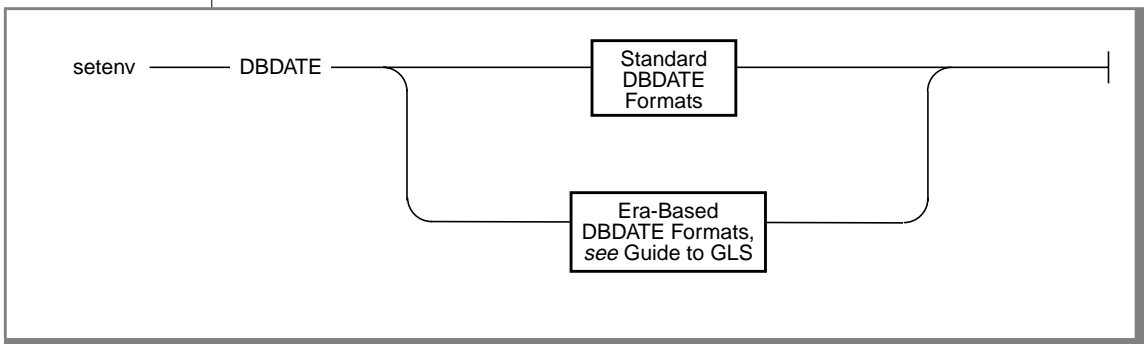
Behavior of DBCENTURY = R

Example data type: DATETIME year to month
Current date: 4/6/1998
User enters: 1/1/1
DBCENTURY = R, Present century algorithm
Present century expansion: 1901-1
Analysis: The present century expansion is used.

DBDATE

The **DBDATE** environment variable specifies the end-user formats of DATE values. End-user formats affect the following situations:

- When you input DATE values, Informix products use the **DBDATE** environment variable to interpret the input. For example, if you specify a literal DATE value in an INSERT statement, Informix database servers expect this literal value to be compatible with the format that **DBDATE** specifies. Similarly, the database server interprets the date that you specify as input to the DATE() function in the format that the **DBDATE** environment variable specifies.
- When you display DATE values, Informix products use the **DBDATE** environment variable to format the output.

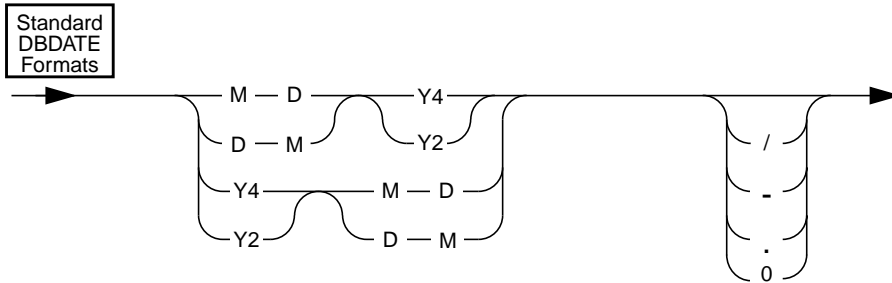


GLS

This section describes standard **DBDATE** formats. For a description of era-based formats, see the [Informix Guide to GLS Functionality](#). ♦

With standard formats, you can specify the following attributes:

- The order of the month, day, and year in a date
- Whether the year should be printed with two digits (Y2) or four digits (Y4)
- The separator between the month, day, and year



- . / are characters that can be used as separators in a date format.
- 0 indicates that no separator is displayed.
- D, M are characters representing the day and the month.
- Y2, Y4 are characters that represent the year and the number of digits in the year.

For the U.S. ASCII English locale, the default setting for **DBDATE** is **MDY4/**, where **M** represents the month, **D** represents the day, **Y4** represents a four-digit year, and slash (/) is a separator (for example, 01/08/1998).

Other acceptable characters for the separator are a hyphen (-), a period (.), or a zero (0). To indicate no separator, use the zero.

The slash (/) appears if you attempt to use a character other than a hyphen, period, or zero as a separator, or if you do not include a separator character in the **DBDATE** definition.

The following table shows a few variations of setting the **DBDATE** environment variable.

Variation	January 8, 1998, appears as:
MDY4/	01/08/1998
DMY2-	08-01-98
MDY4	01/08/1998
Y2DM.	98.08.01
MDY20	010898
Y4MD*	1998/01/08

The formats **Y4MD*** (the asterisk is an unacceptable separator) and **MDY4** (no separator is defined) both display the default (slash) as a separator.

Important: If you use the **Y2** format, understand that the setting of the **DBCENTURY** environment variable affects how the **DATE** values are expanded.

Also, certain routines that **INFORMIX-ESQL/C** calls can use the **DBTIME** variable, rather than **DBDATE**, to set **DATETIME** formats to international specifications. For more information, see the discussion of the **DBTIME** environment variable on [page 3-43](#) and the “[INFORMIX-ESQL/C Programmer’s Manual](#).”

The setting of the **DBDATE** variable takes precedence over that of the **GL_DATE** environment variable, as well as over the default **DATE** formats that **CLIENT_LOCALE** specifies. For information about the **GL_DATE** and **CLIENT_LOCALE** environment variables, see the [Informix Guide to GLS Functionality](#). ♦



GLS

DBDELIMITER

The **DBDELIMITER** environment variable specifies the field delimiter used by the **dbexport** utility and with the **LOAD** and **UNLOAD** statements.

```
setenv DBDELIMITER 'delimiter'
```

delimiter is the field delimiter for unloaded data files.

The delimiter can be any single character, except the characters in the following list:

- Hexadecimal numbers (0 through 9, a through f, A through F)
- Newline or CTRL-J
- The backslash symbol (\)

The vertical bar (|=ASCII 124) is the default. To change the field delimiter to a plus (+), set the **DBDELIMITER** environment variable, as the following example shows:

```
setenv DBDELIMITER '+'
```

DBEDIT

The **DBEDIT** environment variable lets you name the text editor that you want to use to work with SQL statements and command files in DB-Access. If **DBEDIT** is set, the specified text editor is called directly. If **DBEDIT** is not set, you are prompted to specify a text editor as the default for the rest of the session.

```
setenv DBEDIT editor
```

editor is the name of the text editor you want to use.

For most systems, the default text editor is **vi**. If you use another text editor, be sure that it creates flat ASCII files. Some word processors in *document mode* introduce printer control characters that can interfere with the operation of your Informix product.

To specify the EMACS text editor, set the **DBEDIT** environment variable by entering the following command:

```
setenv DBEDIT emacs
```

DBFLTMASK

By default, Informix client applications (including DB-Access utility or any ESQL program) display the floating-point values of data types `FLOAT`, `SMALLFLOAT`, and `DECIMAL` with 16 digits to the right of the decimal point. However, the actual number of decimal digits displayed depends on the size of the character buffer.

To reduce the default number of decimal digits in the display, you can set the **DBFLTMASK** environment variable to the number of digits desired.

```
setenv DBFLTMASK n
```

n is the number of decimal digits that you want the Informix client application to display in the floating-point values. *n* must be smaller than 16, the default number of digits displayed.

DBLANG

The **DBLANG** environment variable specifies the subdirectory of **SINFORMIXDIR** or the full pathname of the directory that contains the compiled message files that an Informix product uses.

```
setenv DBLANG relative_path
           full_path
```

<i>relative_path</i>	is the subdirectory of \$INFORMIXDIR .
<i>full_path</i>	is the full pathname of the directory that contains the compiled message files.

By default, Informix products put compiled messages in a locale-specific subdirectory of the **\$INFORMIXDIR/msg** directory. These compiled message files have the suffix **.iem**. If you want to use a message directory other than **\$INFORMIXDIR/msg**, where, for example, you can store message files that you have created, perform the following steps:

1. Use the **mkdir** command to create the appropriate directory for the message files.

You can make this directory under the directory **\$INFORMIXDIR** or **\$INFORMIXDIR/msg**, or you can make it under any other directory.

2. Set the owner and group of the new directory to **informix** and the access permission for this directory to **755**.
3. Set the **DBLANG** environment variable to the new directory.
If this directory is a subdirectory of **\$INFORMIXDIR** or **\$INFORMIXDIR/msg**, you need only to list the relative path to the new directory. Otherwise, you must specify the full pathname of the directory.
4. Copy the **.iem** files or the message files that you created to the new message directory that **\$DBLANG** specifies.

All the files in the message directory should have the owner and group **informix** and access permission **644**.

Informix products that use the default, U.S. ASCII English, search for message files in the following order:

1. In **\$DBLANG**, if **DBLANG** is set to a full pathname
2. In **\$INFORMIXDIR/msg/\$DBLANG**, if **DBLANG** is set to a relative pathname
3. In **\$INFORMIXDIR/\$DBLANG**, if **DBLANG** is set to a relative pathname
4. In **\$INFORMIXDIR/msg/en_us/0333**

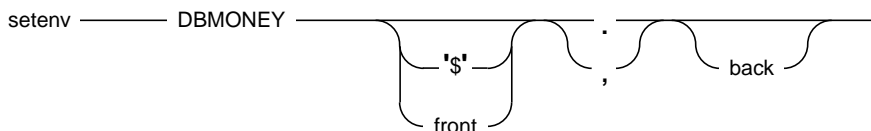
GLS

5. In \$INFORMIXDIR/msg/en_us.8859-1
6. In \$INFORMIXDIR/msg
7. In \$INFORMIXDIR/msg/english

For more information on access paths for messages, see the description of DBLANG in the [Informix Guide to GLS Functionality](#). ♦

DBMONEY

The **DBMONEY** environment variable specifies the display format of monetary values with FLOAT, DECIMAL, or MONEY data types.



- \$ is the default symbol that precedes the MONEY value.
- ,
- .
- back* represents the optional symbol that follows the MONEY value. The *back* symbol can be up to seven characters and can contain any character except an integer, a comma, or a period. If *back* contains a dollar sign (\$), you must enclose the whole string in single quotes (').
- front* is the optional symbol that precedes the MONEY value. The *front* symbol can be up to seven characters and can contain any character except an integer, a comma, or a period. If *front* contains a dollar sign (\$), you must enclose the whole string in single quotes (').

If you use any character except an alphabetic character for *front* or *back*, you must enclose the character in quotes.



When you display MONEY values, Informix products use the **DBMONEY** environment variable to format the output.

Tip: The setting of **DBMONEY** does not affect the internal format of the MONEY column in the database.

If you do not set **DBMONEY**, then MONEY values for the default locale, U.S. ASCII English, are formatted with a dollar sign (\$) preceding the MONEY value, a period (.) separating the integral from the fractional part of the MONEY value, and no *back* symbol. For example, 10050 is formatted as \$100.50.

Suppose you want to represent MONEY values in DM (Deutsche Mark), which uses the currency symbol DM and a comma. Enter the following command to set the **DBMONEY** environment variable:

```
setenv DBMONEY DM,
```

Here, DM is the currency symbol preceding the MONEY value, and a comma separates the integral from the fractional part of the MONEY value. As a result, the amount 10050 is displayed as DM100,50.

For more information about how the **DBMONEY** environment variable handles MONEY formats for nondefault locales, see the [Informix Guide to GLS Functionality](#). ♦

GLS

AD/XP

IDS

DBONPLOAD

The **DBONPLOAD** environment variable specifies the name of the database that the **onpload** utility of the High-Performance Loader uses. If the **DBONPLOAD** environment variable is set, the specified name is the name of the database. If the **DBONPLOAD** environment variable is not set, the default name of the database is **onpload**.

```
setenv _____DBONPLOAD_____ dbname _____|
```

dbname specifies the name of the database that the **onpload** utility uses.

W/D

For example, to specify the name **load_db** as the name of the database, enter the following command:

```
setenv DBONPLOAD load_db
```

The High-Performance Loader is not available for Dynamic Server, Workgroup and Developer Editions. ♦

DBPATH

Use **DBPATH** to identify the database servers that contain databases. The **DBPATH** environment variable also specifies a list of directories (in addition to the current directory) in which DB-Access looks for command scripts (**.sql** files).

The **CONNECT**, **DATABASE**, **START DATABASE**, and **DROP DATABASE** statements use **DBPATH** to locate the database under two conditions:

- If the location of a database is not explicitly stated
- If the database cannot be located in the default server

The **CREATE DATABASE** statement does not use **DBPATH**.

To add a new **DBPATH** entry to existing entries, see [“Modifying the Setting of an Environment Variable in UNIX” on page 3-9](#).

```
setenv DBPATH : //servername
```

servername is the name of an Informix database server on which databases are stored. You cannot reference database files with a **servername**.

DBPATH can contain up to 16 entries. Each entry (*full_pathname*, *servername*, or *servername* and *full_pathname*) must be less than 128 characters. In addition, the maximum length of **DBPATH** depends on the hardware platform on which you set **DBPATH**.

When you access a database with the `CONNECT`, `DATABASE`, `START DATABASE`, or `DROP DATABASE` statement, the search for the database is done first in the directory and/or database server specified in the statement. If no database server is specified, the default database server as set in the `INFORMIXSERVER` environment variable is used.

If the database is not located during the initial search, and if `DBPATH` is set, the database servers and/or directories in `DBPATH` are searched for in the indicated database. The entries to `DBPATH` are considered in order.

Using DBPATH with DB-Access

If you are using DB-Access and you use the **Choose** option of the SQL menu without having already selected a database, you see a list of all the `.sql` files in the directories listed in your `DBPATH`. Once you select a database, the `DBPATH` is not used to find the `.sql` files: Only the `.sql` files in the current working directory are displayed.

Searching Local Directories

Use a pathname without a database server name to have the database server search for `.sql` scripts on your local computer.

In the following example, the `DBPATH` setting causes DB-Access to search for the database files in your current directory and then in Joachim's and Sonja's directories on the local computer:

```
setenv DBPATH /usr/joachim:/usr/sonja
```

As the previous example shows, if the pathname specifies a directory name but not a database server name, the directory is sought on the computer that runs the default database server that the `INFORMIXSERVER` environment variable specifies (see [page 3-54](#)). For instance, with the previous example, if `INFORMIXSERVER` is set to `quality`, the `DBPATH` value is *interpreted*, as the following example shows, where the double slash precedes the database server name:

```
setenv DBPATH //quality/usr/joachim://quality/usr/sonja
```

Searching Networked Computers for Databases

If you use more than one database server, you can set **DBPATH** to explicitly contain the database server and/or directory names that you want to search for databases. For example, if **INFORMIXSERVER** is set to **quality** but you also want to search the **marketing** database server for **/usr/joachim**, set **DBPATH** as the following example shows:

```
setenv DBPATH //marketing/usr/joachim:/usr/sonja
```

Specifying a Servername

You can set **DBPATH** to contain only database server names. This setting allows you to locate only databases and not locate command files.

The database administrator must include each database server mentioned by **DBPATH** in the **\$INFORMIXDIR/etc/sqlhosts** file. For information on communication-configuration files and dbservernames, see your [Administrator's Guide](#).

For example, if **INFORMIXSERVER** is set to **quality**, you can search for a database first on the **quality** database server and then on the **marketing** database server by setting **DBPATH** as the following example shows:

```
setenv DBPATH //marketing
```

If you use DB-Access in this example, the names of all the databases on the **quality** and **marketing** database servers are displayed with the **Select** option of the **DATABASE** menu.

UNIX

DBPRINT

The **DBPRINT** environment variable specifies the printing program that you want to use.

```
setenv DBPRINT program
```

program

names any command, shell script, or UNIX utility that handles standard ASCII input.

The default program is found in one of two places:

- For most BSD UNIX systems, the default program is **lpr**.
- For UNIX System V, the default program is usually **lp**.

Enter the following command to set the **DBPRINT** environment variable to specify the **myprint** print program:

```
setenv DBPRINT myprint
```

DBREMOTECMD

You can set the **DBREMOTECMD** environment variable to override the default remote shell used when you perform remote tape operations with the database server.

The **DBREMOTECMD** environment variable is not available for Dynamic Server, Workgroup and Developer Editions. ♦

You can set the **DBREMOTECMD** environment variable with either a simple command or the full pathname. If you use the full pathname, the database server searches your **PATH** for the specified command.

[illegible]

command is the command to override the default remote shell.

pathname is the pathname to override the default remote shell.

Informix highly recommends the use of the full pathname syntax on the interactive UNIX platform to avoid problems with similarly named programs in other directories and possible confusion with the *restricted shell* (**/usr/bin/rsh**).

Enter the following command to set the **DBREMOTECMD** environment variable for a simple command name:

```
setenv DBRFMOTFCMD rcmd
```

Enter the following command to set the **DBREMOTECMD** environment variable to specify the full pathname:

```
setenv DBREMOTECMD /usr/bin/remsh
```

For more information on **DBREMOTECMD**, see the discussion in your [Archive and Backup Guide](#) about how to use remote tape devices with your database server for archives, restores, and logical-log backups.

DBSPACETEMP

You can set your **DBSPACETEMP** environment variable to specify the dbspaces in which temporary tables are to be built.

You can specify multiple dbspaces to spread temporary space across any number of disks.

```
setenv _____ DBSPACETEMP _____
```

punct
temp_dspace

punct can be either colons or commas.
temp_dspace is a valid existing temporary dbspace.

The **DBSPACETEMP** environment variable overrides the default dbspaces that the **DBSPACETEMP** configuration parameter specifies in the configuration file for your database server.

For example, you might set the **DBSPACETEMP** environment variable with the following command:

```
setenv DBSPACETEMP sorttmp1:sorttmp2:sorttmp3
```

Separate the dbspace entries with either colons or commas. The number of dbspaces is limited by the maximum size of the environment variable, as defined by your operating system. Your database server does not create a dbspace specified by the environment variable if the dbspace does not exist.

The two classes of temporary tables are explicit temporary tables that the user creates and implicit temporary tables that the database server creates. Use the **DBSPACETEMP** environment variable to specify the dbspaces for both types of temporary tables.

If you create an explicit temporary table with the `CREATE TEMP TABLE` statement and do not specify a `dbspace` for the table either in the `IN dbspace` clause or in the `FRAGMENT BY` clause, the database server uses the settings in the **DBSPACETEMP** environment variable to determine where to create the table. If the **DBSPACETEMP** environment variable is not set, the database server uses the `ONCONFIG` parameter **DBSPACETEMP**. If this parameter is not set, the database server creates the temporary table in the same `dbspace` where the database resides.

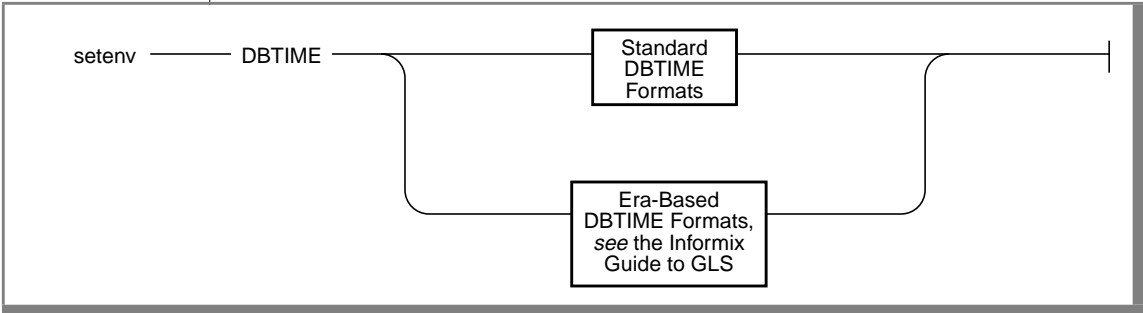
If you create an explicit temporary table with the `SELECT INTO TEMP` statement, the database server uses the settings in the **DBSPACETEMP** environment variable to determine where to create the table. If the **DBSPACETEMP** environment variable is not set, the database server uses the `ONCONFIG` parameter **DBSPACETEMP**. If this parameter is not set, the database server creates the temporary table in the root `dbspace`.

The database server creates implicit temporary tables for its own use while executing join operations, `SELECT` statements with the `GROUP BY` clause, `SELECT` statements with the `ORDER BY` clause, and index builds. When it creates these implicit temporary tables, the database server uses disk space for writing the temporary data, in the following order:

1. The operating-system directory or directories that the environment variable **PSORT_DBTEMP** specifies, if it is set. ♦
2. The `dbspace` or `dbspaces` that the environment variable **DBSPACETEMP** specifies, if it is set.
3. The `dbspace` or `dbspaces` that the `ONCONFIG` parameter **DBSPACETEMP** specifies.
4. The operating-system file space in `/tmp` (UNIX) or `%temp%` (Windows NT).

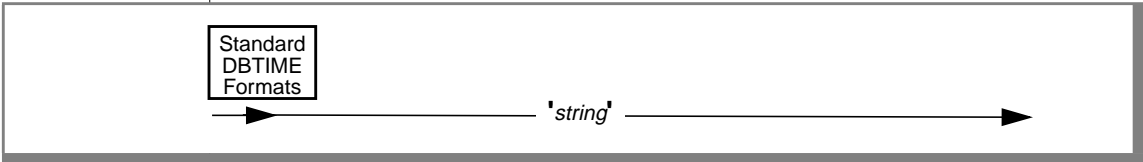
DBTIME

The **DBTIME** environment variable specifies the end-user formats of DATETIME values for a set of SQL API library functions.



You can set the **DBTIME** environment variable to manipulate DATETIME formats so that the formats conform more closely to various international or local TIME conventions. **DBTIME** takes effect only when you call certain INFORMIX-ESQL/C DATETIME routines; otherwise, use the **DBDATE** environment variable. (For details, see the [INFORMIX-ESQL/C Programmer's Manual](#).)

You can set **DBTIME** to specify the exact format of an input/output (I/O) DATETIME string field with the formatting directives described in the following list. Otherwise, the behavior of the DATETIME formatting routine is undefined.



string The formatting directives that you can use are described in the following list:

- %b is replaced by the abbreviated month name.
- %B is replaced by the full month name.

%d	is replaced by the day of the month as a decimal number [01,31].
%Fn	is replaced by the value of the fraction with precision that the integer <i>n</i> specifies. The default value of <i>n</i> is 2; the range of <i>n</i> is $0 \leq n \leq 5$.
%H	is replaced by the hour (24-hour clock).
%I	is replaced by the hour (12-hour clock).
%M	is replaced by the minute as a decimal number [00,59].
%m	is replaced by the month as a decimal number [01,12].
%p	is replaced by A.M. or P.M. (or the equivalent in the local standards).
%S	is replaced by the second as a decimal number [00,59].
%y	is replaced by the year as a four-digit decimal number. If the user enters a two-digit value, the format of this value is affected by the setting of the DBCENTURY environment variable. If DBCENTURY is not set, then the current century is used for the century digits.
%Y	is replaced by the year as a four-digit decimal number. User must enter a four-digit value.
%%	is replaced by % (to allow % in the format string).

(2 of 2)

For example, consider how to convert a DATETIME YEAR TO SECOND to the following ASCII string format:

```
Mar 21, 1998 at 16 h 30 m 28 s
```

Set **DBTIME** as the following list shows:

```
setenv DBTIME '%b %d, %Y at %H h %M m %S s'
```

The default **DBTIME** produces the conventional ANSI SQL string format that the following line shows:

```
1998-03-21 16:30:28
```

Set the default **DBTIME** as the following example shows:

```
setenv DBTIME '%Y-%m-%d %H:%M:%S'
```

An optional field width and precision specification can immediately follow the percent (%) character; it is interpreted as the following list describes:

- [-|0]w** where *w* is a decimal digit string specifying the minimum field width. By default, the value is right justified with spaces on the left. If **-** is specified, it is left justified with spaces on the right. If **0** is specified, it is right justified and padded with zeros on the left.
- .p** where *p* is a decimal digit string specifying the number of digits to appear for **d**, **H**, **I**, **m**, **M**, **S**, **y**, and **Y** conversions, and the maximum number of characters to be used for **b** and **B** conversions. A precision specification is significant only when converting a **DATETIME** value to an ASCII string and not vice versa.

When you use field width and precision specifications, the following limitations apply:

- If a conversion specification supplies fewer digits than a precision specifies, it is padded with leading zeros.
- If a conversion specification supplies more characters than a precision specifies, excess characters are truncated on the right.
- If no field width or precision is specified for **d**, **H**, **I**, **m**, **M**, **S**, or **y** conversions, a default of **0.2** is used. A default of **0.4** is used for **Y** conversions.

The **F** conversion does not follow the field width and precision format conversions that are described earlier.

For related information, see the discussion of **DBDATE** on [page 3-29](#).

DBUPSPACE

The **DBUPSPACE** environment variable lets you specify and constrain the amount of system disk space that the UPDATE STATISTICS statement can use when trying to simultaneously construct multiple column distributions.

```
setenv _____ DBUPSPACE _____ value _____
```

value represents a disk space amount in kilobytes.

For example, to set **DBUPSPACE** to 2,500 kilobytes, enter the following command:

```
setenv DBUPSPACE 2500
```

Once you set this value, then the database server can use no more than 2,500 kilobytes of disk space during the execution of an UPDATE STATISTICS statement. If a table requires 5 megabytes of disk space for sorting, then UPDATE STATISTICS accomplishes the task in two passes; the distributions for one half of the columns are constructed with each pass.

If you try to set **DBUPSPACE** to any value less than 1,024 kilobytes, it is automatically set to 1,024 kilobytes, but no error message is returned. If this value is not large enough to allow more than one distribution to be constructed at a time, at least one distribution is done, even if the amount of disk space required for the one is greater than specified in **DBUPSPACE**.

DELIMIDENT

The **DELIMIDENT** environment variable specifies that strings set off by double quotes are delimited identifiers.

```
setenv _____ DELIMIDENT _____
```

You can use delimited identifiers to specify identifiers that are identical to reserved keywords, such as `TABLE` or `USAGE`. You can also use them to specify database identifiers that contain nonalpha characters, but you cannot use them to specify storage identifiers that contain non-alpha characters. Note that database identifiers are names for database objects such as tables and columns, and storage identifiers are names for storage objects such as dbspaces and partition blobs.

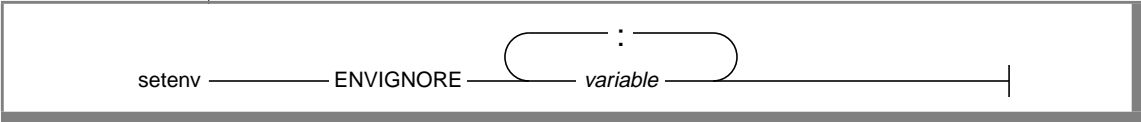
Delimited identifiers are case sensitive.

To use delimited identifiers, applications in ESQL/C must set the **DELIMIDENT** environment variable at compile time and execute time.

UNIX

ENVIGNORE

Use the ENVIGNORE environment variable to deactivate specified environment variable entries in the common (shared) and private environment-configuration files, **informix.rc** and **.informix** respectively.



variable is the list of environment variables that you want to deactivate.

For example, to ignore the **DBPATH** and **DBMONEY** entries in the environment-configuration files, enter the following command:

```
setenv ENVIGNORE DBPATH:DBMONEY
```

The common environment-configuration file is stored in **\$INFORMIXDIR/etc/informix.rc**. The private environment-configuration file is stored in the user’s home directory as **.informix**. For information on creating or modifying an environment-configuration file, see [“Environment Variables in an Environment-Configuration File” on page 3-6](#).

ENVIGNORE cannot be set in an environment-configuration file.

FET_BUF_SIZE

The FET_BUF_SIZE environment variable lets you override the default setting for the size of the fetch buffer for all data except blobs. When set, FET_BUF_SIZE is effective for the entire environment.



n represents the size of the buffer in bytes.

When set to a valid value, the environment variable overrides the previously set value. The default setting for the fetch buffer is dependent on row size.

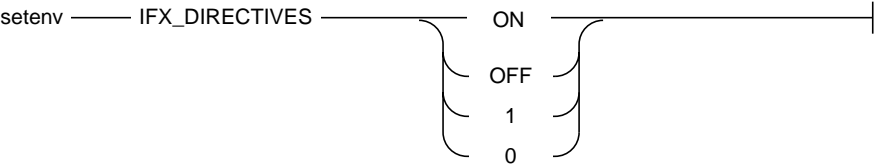
If the buffer size is set to less than the default size or is out of the range of the small integer value, no error is raised. The new buffer size is ignored.

For example, to set a buffer size to 5,000 bytes, set the **FET_BUF_SIZE** environment variable by entering the following command:

```
setenv FET_BUF_SIZE 5000
```

IFX_DIRECTIVES

The **IFX_DIRECTIVES** environment variable setting determines whether the optimizer allows query optimization directives from within a query. The **IFX_DIRECTIVES** environment variable is set on the client. You can use either **ON** and **OFF** or **1** and **0** to set the environment variable.



- | | |
|-----|-----------------------------------|
| ON | Optimizer directives accepted |
| OFF | Optimizer directives not accepted |
| 1 | Optimizer directives accepted |
| 0 | Optimizer directives not accepted |

The setting of the **IFX_DIRECTIVES** environment variable overrides the **DIRECTIVES onconfig** parameter value that is set for the server. If the **IFX_DIRECTIVES** environment variable is not set, however, then all client sessions will inherit the server configuration for directives that the **DIRECTIVES onconfig** parameter determines. The default setting for the **DIRECTIVES onconfig** parameter is **ON**.



If **INFORMIXC** is not set, the default compiler is cc.

Tip: On Windows NT, you pass either **-mcc** or **-bcc** options to the **esql** preprocessor to use either the Microsoft or Borland C compilers.

compiler is the name of the C compiler.
pathname is the full pathname of the C compiler.

For example, to specify the GNU C compiler, enter the following command:

```
setenv INFORMIXC gcc
```

The setting is required only during the C compilation stage.

INFORMIXCONRETRY

The **INFORMIXCONRETRY** environment variable specifies the maximum number of additional connection attempts that should be made to each server by the client during the time limit that the **INFORMIXCONTIME** environment variable specifies.

value represents the number of connection attempts to each server.

For example, enter the following command to set **INFORMIXCONRETRY** to three additional connection attempts (after the initial attempt):

```
setenv INFORMIXCONRETRY 3
```

The default value for **INFORMIXCONRETRY** is one retry after the initial connection attempt. The **INFORMIXCONTIME** setting, described in the following section, takes precedence over the **INFORMIXCONRETRY** setting.

INFORMIXCONTIME

The **INFORMIXCONTIME** environment variable lets you specify that an SQL **CONNECT** statement should keep trying for at least the given number of seconds before returning an error.

You might encounter connection difficulties related to system or network load problems. For instance, if the database server is busy establishing new SQL client threads, some clients might fail because the server cannot issue a network function call fast enough. The **INFORMIXCONTIME** and **INFORMIXCONRETRY** environment variables let you configure your client-side connection capability to retry the connection instead of returning an error.

```
setenv _____ INFORMIXCONTIME _____ value _____
```

value represents the minimum number of seconds spent in attempts to establish a connection to a server.

For example, enter the following command to set **INFORMIXCONTIME** to 60 seconds:

```
setenv INFORMIXCONTIME 60
```

If **INFORMIXCONTIME** is set to 60 and **INFORMIXCONRETRY** is set to 3, as these examples show, attempts to connect to the server (after the initial attempt at 0 seconds) will be made at 20, 40, and 60 seconds, if necessary, before aborting. This 20-second interval is the result of **INFORMIXCONTIME** divided by **INFORMIXCONRETRY**.

If execution of the **CONNECT** statement involves searching **DBPATH**, the following rules apply:

- All appropriate servers in the **DBPATH** setting are accessed at least once, even though the **INFORMIXCONTIME** value might be exceeded. Thus, the **CONNECT** statement might take longer than the **INFORMIXCONTIME** time limit to return an error that indicates connection failure or that the database was not found.
- The **INFORMIXCONRETRY** value specifies the number of additional connections that should be attempted for each server entry in **DBPATH**.
- The **INFORMIXCONTIME** value is divided among the number of server entries specified in **DBPATH**. Thus, if **DBPATH** contains numerous servers, you should increase the **INFORMIXCONTIME** value accordingly. For example, if **DBPATH** contains three entries, to spend at least 30 seconds attempting each connection, set **INFORMIXCONTIME** to 90.

The default value for **INFORMIXCONTIME** is 15 seconds. The setting for **INFORMIXCONTIME** takes precedence over the **INFORMIXCONRETRY** setting. Retry efforts could end after the **INFORMIXCONTIME** value is exceeded, but before the **INFORMIXCONRETRY** value is reached.

INFORMIXDIR

The **INFORMIXDIR** environment variable specifies the directory that contains the subdirectories in which your product files are installed. You must always set the **INFORMIXDIR** environment variable. Verify that the **INFORMIXDIR** environment variable is set to the full pathname of the directory in which you installed your database server. If you have multiple versions of a database server, set **INFORMIXDIR** to the appropriate directory name for the version that you want to access. For information about when to set the **INFORMIXDIR** environment variable, see your [Installation Guide](#).

setenv _____ INFORMIXDIR _____ *pathname* _____

pathname is the directory path where the product files are installed.

IDS

UNIX

Enter the following command to set the **INFORMIXDIR** environment variable to the desired installation directory:

```
setenv INFORMIXDIR /usr/informix
```

INFORMIXKEYTAB

The **INFORMIXKEYTAB** environment variable specifies the location of the **keytab** file. The **keytab** file contains authentication information that database servers and clients access at connection time, if they use the DCE-GSS communications support module (CSM). It contains key tables that store keys, each of which contains a principal name (for a database server name or a user name), type, version, and value. The database server uses the **keytab** file to find the key to register the server and to acquire a credential for it. A client application uses the key if the user did not do **dce_login** with the current operating-system user name (which is the same as the DCE principle name) or did not explicitly provide credential.

```
setenv _____ +_____ pathname
```

pathname specifies the full path of the **keytab** file.

For example, the following command specifies that the name and location of the **keytab** file is **/usr/myfiles/mykeytab**:

```
setenv INFORMIXKEYTAB /usr/myfiles/mykeytab
```

For more information about the DCE-GSS communications support module, see your [Administrator's Guide](#).

IDS

INFORMIXOPCACHE

The **INFORMIXOPCACHE** environment variable lets you specify the size of the memory cache for the staging-area blob space of the client application.

W/D

The **INFORMIXOPCACHE** environment variable is not available for Informix Dynamic Server, Workgroup and Developer Editions. ♦

setenv

kilobytes

kilobytes specifies the value you set for the optical memory cache.

You set the **INFORMIXOPCACHE** environment variable by specifying the size of the memory cache in kilobytes. The specified size must be equal to or smaller than the size of the system-wide configuration parameter, **OPCACHEMAX**. If you do not set the **INFORMIXOPCACHE** environment variable, the default cache size is 128 kilobytes or the size specified in the configuration parameter **OPCACHEMAX**. The default for **OPCACHEMAX** is 128 kilobytes. If you set **INFORMIXOPCACHE** to a value of 0, Optical Subsystem does not use the cache.

INFORMIXSERVER

The **INFORMIXSERVER** environment variable specifies the default database server to which an explicit or implicit connection is made by an SQL API client or the DB-Access utility.

The database server can be either local or remote. You must always set **INFORMIXSERVER** before you use an Informix product.

setenv

INFORMIXSERVER

dbservername

dbservername is the name of the default database server.



AD/XP

The value of **INFORMIXSERVER** must correspond to a valid *dbservername* entry in the `$INFORMIXDIR/etc/sqlhosts` file on the computer running the application. The *dbservername* must be specified using lowercase characters and cannot exceed 18 characters. For example, specify the **coral** database server as the default for connection by entering the following command:

```
setenv INFORMIXSERVER coral
```

INFORMIXSERVER specifies the database server to which an application connects if the **CONNECT DEFAULT** statement is executed. It also defines the database server to which an initial implicit connection is established if the first statement in an application is not a **CONNECT** statement.

Important: You must set **INFORMIXSERVER** even if the application or DB-Access does not use implicit or explicit default connections.

For Dynamic Server with AD and XP Options, the **INFORMIXSERVER** environment variable specifies the name of a *dbserver* group. To specify a *coserver* name, use the following format:

```
dbservername.coserver_number
```

In the *coserver* name, *dbservername* is the value that you assigned to the **DBSERVERNAME** configuration parameter when you prepared the **ONCONFIG** configuration file, and *coserver_number* is the value that you assigned to the **COSERVER** configuration parameter for the connection *coserver*.

Strictly speaking, **INFORMIXSERVER** is not required for initialization. However, if **INFORMIXSERVER** is not set, Dynamic Server with AD and XP Options does not build the **sysmaster** tables. ♦

UNIX

INFORMIXSHMBASE

The **INFORMIXSHMBASE** environment variable affects only client applications connected to Informix databases that use the IPC shared-memory (**ipcshm**) communication protocol.

Important: Resetting **INFORMIXSHMBASE** requires a thorough understanding of how the application uses memory. Normally you do not reset **INFORMIXSHMBASE**.



Use **INFORMIXSHMBASE** to specify where shared-memory communication segments are attached to the client process so that client applications can avoid collisions with other memory segments that the application uses. If you do not set **INFORMIXSHMBASE**, the memory address of the communication segments defaults to an implementation-specific value such as 0x800000.

```
setenv INFORMIXSHMBASE value
```

value is used to calculate the memory address.

The database server calculates the memory address where segments are attached by multiplying the value of **INFORMIXSHMBASE** by 1,024. For example, to set the memory address to the value 0x800000, set the **INFORMIXSHMBASE** environment variable by entering the following command:

```
setenv INFORMIXSHMBASE 8192
```

For more information, see your [Administrator's Guide](#).

INFORMIXSQLHOSTS

The **INFORMIXSQLHOSTS** environment variable specifies the full pathname to the place where client-database server connectivity information is stored. On a UNIX system, by default, this environment variable points to the **\$INFORMIXDIR/etc/sqlhosts** file. For example, to specify that the client or database server will look for connectivity information in the **mysqlhosts** file in the **/work/envt** directory, enter the following command:

```
setenv INFORMIXSQLHOSTS /work/envt/mysqlhosts
```

When the **INFORMIXSQLHOSTS** environment variable is set, the client or database server looks in the specified file for connectivity information. When the **INFORMIXSQLHOSTS** environment variable is not set, the client or database server looks in the **\$INFORMIXDIR/etc/sqlhosts** file.

```
setenv _____ | _____ pathname _____|
```

pathname specifies the full pathname and filename of the file that contains connectivity information.

On a Windows NT system, by default, this environment variable points to the computer whose registry contains the **SQLHOSTS** subkey. To specify that the client or database server will look for connectivity information on a computer named **arizona**, enter the following command:

```
set INFORMIXSQLHOSTS = \\arizona
```

For a description of the **SqlHosts** information, see your [Administrator's Guide](#).

INFORMIXSTACKSIZE

INFORMIXSTACKSIZE specifies the stack size (in kilobytes) that the database server uses for a particular client session.

Use **INFORMIXSTACKSIZE** to override the value of the **ONCONFIG** parameter **STACKSIZE** for a particular application or user.

```
setenv _____ INFORMIXSTACKSIZE _____ value _____|
```

value is the stack size for SQL client threads in kilobytes.

For example, to decrease the **INFORMIXSTACKSIZE** to 20 kilobytes, enter the following command:

```
setenv INFORMIXSTACKSIZE 20
```



UNIX

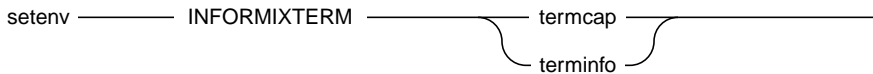
If **INFORMIXSTACKSIZE** is not set, the stack size is taken from the database server configuration parameter **STACKSIZE**, or it defaults to a platform-specific value. The default stack size value for the primary thread for an SQL client is 32 kilobytes for nonrecursive database activity.

Warning: For specific instructions on setting this value, see your “[Administrator’s Guide](#).” If you incorrectly set the value of **INFORMIXSTACKSIZE**, it can cause the database server to fail.

INFORMIXTERM

The **INFORMIXTERM** environment variable specifies whether DB-Access should use the information in the **termcap** file or the **terminfo** directory.

The **termcap** file and **terminfo** directory determine terminal-dependent keyboard and screen capabilities, such as the operation of function keys, color and intensity attributes in screen displays, and the definition of window borders and graphic characters.



If **INFORMIXTERM** is not set, the default setting is **termcap**. When DB-Access is installed on your system, a **termcap** file is placed in the **etc** subdirectory of **\$INFORMIXDIR**. This file is a superset of an operating-system **termcap** file.

You can use the **termcap** file that Informix supplies, the system **termcap** file, or a **termcap** file that you create. You must set the **TERMCAP** environment variable if you do not use the default **termcap** file. For information on setting the **TERMCAP** environment variable, see [page 3-69](#).

The **terminfo** directory contains a file for each terminal name that has been defined. The **terminfo** setting for **INFORMIXTERM** is supported only on computers that provide full support for the UNIX System V **terminfo** library. For details, see the machine-notes file for your product.

IDS

UNIX



INF_ROLE_SEP

The `INF_ROLE_SEP` environment variable configures the security feature of role separation when the database server is installed. Role separation enforces separating administrative tasks that different people who are involved in running and auditing the database server perform.

Tip: To enable role separation for database servers on Windows NT, choose the role separation option during installation.

If `INF_ROLE_SEP` is set, role separation is implemented and a separate group is specified to serve each of the following responsibilities: the database system security officer (DBSSO), the audit analysis officer (AAO), and the standard user. If `INF_ROLE_SEP` is not set, user **informix** (the default) can perform all administrative tasks.

setenv

n is any positive integer.

For more information about the security feature of role separation, see the [Trusted Facility Manual](#). To learn how to configure role separation when you install your database server, see your [Installation Guide](#).

NODEFDAC

When the `NODEFDAC` environment variable is set to `yes`, it prevents default table privileges (Select, Insert, Update, and Delete) from being granted to PUBLIC when a new table is created in a database that is not ANSI compliant. If you do not set the `NODEFDAC` variable, it is, by default, set to `no`.

setenv

N

no

<i>yes</i>	prevents default table privileges from being granted to PUBLIC on new tables in a database that is not ANSI compliant. This setting also prevents the Execute privilege for a new stored procedure from being granted to PUBLIC when the stored procedure is created in owner mode.
<i>no</i>	allows default table privileges to be granted to PUBLIC. Also allows the Execute privilege on a new stored procedure to be granted to PUBLIC when the stored procedure is created in owner mode.

ONCONFIG

The ONCONFIG environment variable specifies the name of the active file that holds configuration parameters for the database server. This file is read as input during the initialization procedure. After you prepare the ONCONFIG configuration file, set the ONCONFIG environment variable to the name of the file.

If the ONCONFIG environment variable is not present, the database server uses configuration values from the file \$INFORMIXDIR/etc/onconfig.



filename is the name of a file in \$INFORMIXDIR/etc that contains the configuration parameters for your database.

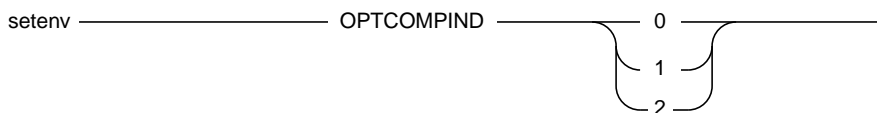
To prepare the ONCONFIG file, make a copy of the **onconfig.std** file and modify the copy. Informix recommends that you name the ONCONFIG file so that it can easily be related to a specific database server. If you have multiple instances of a database server, each instance *must* have its own uniquely named ONCONFIG file.

To prepare the **ONCONFIG** file for Dynamic Server with AD and XP Options, make a copy of the **onconfig.std** file if you are using a single coserver configuration or make a copy of the **onconfig.xps** file if you are using a multiple coserver configuration. You can use the **onconfig.std** file for a multiple coserver configuration, but you would have to add additional keywords and configuration parameters such as **END**, **NODE**, and **COSERVER**, which have already been provided for you in the **onconfig.xps** file. ♦

If you do not set the **ONCONFIG** environment variable, the default filename is **onconfig**. For more information, see your [Administrator's Guide](#).

OPTCOMPIND

You can set the **OPTCOMPIND** environment variable so that the optimizer can select the appropriate join method.



- 0 A nested-loop join is preferred, where possible, over a sort-merge join or a hash join.
- 1 When the transaction isolation mode is *not* Repeatable Read, the optimizer behaves as in setting 2; otherwise, the optimizer behaves as in setting 0.
- 2 Nested-loop joins are not necessarily preferred. The optimizer bases its decision purely on costs, regardless of transaction isolation mode.

When the **OPTCOMPIND** environment variable is not set, the database server uses the value specified for the **ONCONFIG** configuration parameter **OPTCOMPIND**. When neither the environment variable nor the configuration parameter is set, the default value is 2.

For more information on the **ONCONFIG** configuration parameter **OPTCOMPIND**, see your [Administrator's Guide](#). For more information on the different join methods for your database server that the optimizer uses, see your [Performance Guide](#).

PATH

The **PATH** environment variable tells the operating system where to search for executable programs. You must include the directory that contains your Informix product to your **PATH** environment variable before you can use the product. This directory should appear before **\$INFORMIXDIR/bin**, which you must also include.

The UNIX **PATH** environment variable tells the shell which directories to search for executable programs. You must add the directory that contains your Informix product to your **PATH** environment variable before you can use the product.

```
setenv PATH $PATH: pathname
```

pathname specifies the search path for the executables.

You can specify the correct search path in various ways. Be sure to include a colon between the directory names.

For additional information about how to modify your path, see [“Modifying the Setting of an Environment Variable in UNIX” on page 3-9](#).

PDQPRIORITY

With Informix Dynamic Server, the **PDQPRIORITY** environment variable determines the degree of parallelism that the database server uses and affects how the database server allocates resources, including memory, processors, and disk reads. ♦

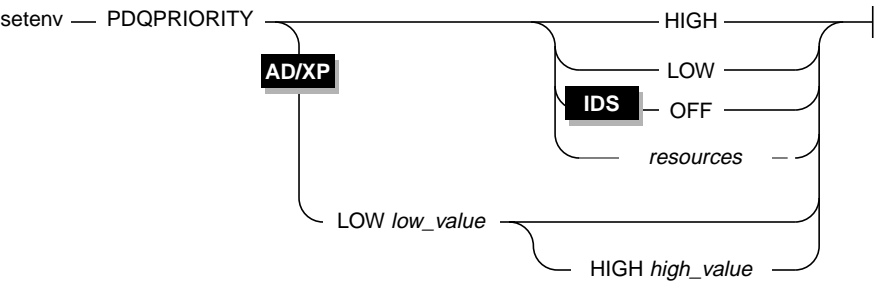
This environment variable is not applicable to Informix Dynamic Server, Workgroup and Developer Editions. ♦

IDS

W/D

AD/XP

In Dynamic Server with AD and XP Options, the **PDQPRIORITY** environment variable only determines the allocation of memory resources. ♦



HIGH	When the database server allocates resources among all users, it gives as many resources as possible to the query.
LOW	Data is fetched from fragmented tables in parallel, but no other parallelism is used.
OFF	PDQ processing is turned off.
resources	Integer value that specifies the query priority level and the amount of resources the database server uses to process the query. Value must be -1, 0, or in the range 0 to 100. Value -1 is the same as DEFAULT. Value 0 is the same as OFF. Value 1 is the same as LOW.
low_value	Integer values that establish the minimum and maximum value of the priority
high_value	

When the **PDQPRIORITY** environment variable is not set, the default value is OFF.

When the environment variable is set to HIGH, the database server determines an appropriate value to use for **PDQPRIORITY** based on several criteria, including the number of available processors, the fragmentation of tables queried, the complexity of the query, and so on.

Usually the more resources a database server uses, the better its performance for a given query, but using too many resources can cause contention among the resources and also take away resources from other queries, which results in degraded performance.

An application can override the setting of the environment variable when it issues the SQL statement SET PDQPRIORITY, which is described in the [Informix Guide to SQL: Syntax](#).

IDS

PLCONFIG

The **PLCONFIG** environment variable specifies the name of the configuration file that the High-Performance Loader uses. This configuration file must reside in the **\$INFORMIXDIR/etc** directory. If the **PLCONFIG** environment variable is not set, the default configuration file is the **\$INFORMIXDIR/etc/plconfig** file.

```
setenv _____ PLCONFIG _____ filename _____
```

filename specifies the simple filename of the configuration file that the High-Performance Loader uses.

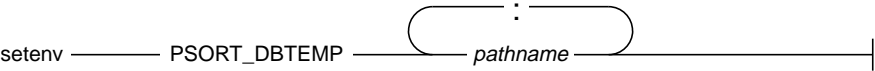
For example, to specify the **\$INFORMIXDIR/etc/custom.cfg** file as the configuration file for the High-Performance Loader, enter the following command:

```
setenv PLCONFIG custom.cfg
```

PSORT_DBTEMP

The **PSORT_DBTEMP** environment variable specifies a directory or directories where the database server writes the temporary files it uses when it performs a sort.

The database server uses the directory that **PSORT_DBTEMP** specifies even if the environment variable **PSORT_NPROCS** is not set.



pathname is the name of the UNIX directory used for intermediate writes during a sort.

To set the **PSORT_DBTEMP** environment variable to specify the directory (for example, `/usr/leif/tempSORT`), enter the following command:

```
setenv PSORT_DBTEMP /usr/leif/tempSORT
```

For maximum performance, specify directories that reside in file systems on different disks.

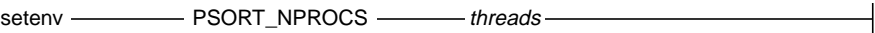
You might also want to consider setting the environment variable **DBSPACETEMP** to place temporary files used in sorting in dbspaces rather than operating-system files. See the discussion of the **DBSPACETEMP** environment variable on [page 3-41](#).

For additional information about the **PSORT_DBTEMP** environment variable, see your [Administrator's Guide](#) and your [Performance Guide](#).

PSORT_NPROCS

The **PSORT_NPROCS** environment variable enables the database server to improve the performance of the parallel-process sorting package by allocating more threads for sorting.

Before the sorting package performs a parallel sort, make sure that the database server has enough memory for the sort.



threads specifies the maximum number of threads to be used to sort a query. The maximum value of *threads* is 10.

Use the following command to set the **PSORT_NPROCS** environment variable to 4:

```
setenv PSORT_NPROCS 4
```

To maximize the effectiveness of the parallel sort, set **PSORT_NPROCS** to the number of available processors in the hardware.

To disable parallel sorting, enter the following command:

```
unsetenv PSORT_NPROCS
```



Tip: If the **PDQPRIORITY** environment variable is not set, the database server allocates the minimum amount of memory to sorts. This minimum memory is insufficient to start even two sort threads. If you have not set the **PDQPRIORITY** environment variable, check the available memory before you perform a large-scale sort (such as an index build) and make sure that you have enough memory.

Default Values for Ordinary Sorts

If the **PSORT_NPROCS** environment variable is set, the database server uses the specified number of sort threads as an upper limit for ordinary sorts. If **PSORT_NPROCS** is not set, parallel sorting does not take place. The database server uses one thread for the sort. If **PSORT_NPROCS** is set to 0, the database server uses three threads for the sort.

Default Values for Attached Indexes

The default number of threads is different for attached indexes.

If the **PSORT_NPROCS** environment variable is set, you get the specified number of sort threads for each fragment of the index that is being built.

If the **PSORT_NPROCS** environment variable is not set, or if it is set to 0, you get two sort threads for each fragment of the index unless you have a single-CPU virtual processor. If you have a single-CPU virtual processor, you get one sort thread for each fragment of the index.

For additional information about the **PSORT_NPROCS** environment variable, see your [Administrator's Guide](#) and your [Performance Guide](#).

IDS

UNIX

SQLEXEC

The **SQLEXEC** environment variable specifies the location of the Version 6.0 or later relay-module executable that allows a Version 5.0 or earlier client to communicate with a local Version 6.0 or later Informix Dynamic Server. Therefore, set **SQLEXEC** only if you want to establish communication between a Version 5.0 or earlier client and a Version 6.0 or later database server.

```
setenv SQLEXEC pathname
```

pathname specifies the pathname for the relay module.

To set **SQLEXEC** to specify the full pathname of the relay module, which is in the **lib** subdirectory of your **\$INFORMIXDIR** directory, enter the following command:

```
setenv SQLEXEC $INFORMIXDIR/lib/sqlrm
```

If you set the **SQLEXEC** environment variable on the C shell command line, you must include curly braces around the existing **INFORMIXDIR**, as the following command shows:

```
setenv SQLEXEC ${INFORMIXDIR}/lib/sqlrm
```

Important: This environment variable functions differently in Version 5.0 and Version 6.0 and later Informix products. For details of Version 5.0 functionality, refer to Chapter 4 of the December 1991 release of this manual.

Tip: Version 5.0 or earlier clients connect to Informix database servers on Windows NT using **INFORMIX-NET PC**.

For information on the relay module, see your [Administrator's Guide](#).



IDS

UNIX

SQLRM

Version 5.0 or earlier clients connect to Informix database servers on Windows NT using INFORMIX-NET PC.

In Version 6.0 and later, if the system administrator is configuring a client/server environment in which a Version 5.0 SQL API client accesses a local Version 6.0 or later database server, the **SQLRM** environment variable must be *unset* before **SQLEXEC** can be used to spawn a Version 6.0 or later relay module.

To unset **SQLRM**, enter the following command:

```
unsetenv SQLRM
```



Important: This environment variable functions differently in Version 5.0 and Version 6.0 and later Informix products. For details of Version 5.0 functionality, refer to Chapter 4 of the December 1991 release of this manual.



Tip: Version 5.0 or earlier clients connect to Informix database servers on Windows NT using INFORMIX-NET PC.

For information on the relay module, see your [Administrator's Guide](#).

IDS

UNIX

SQLRMDIR

In Version 6.0 and later, if the DBA is configuring a client/server environment in which a Version 5.0 SQL API client accesses a local Version 6.0 or later database server, the **SQLRMDIR** environment variable must be *unset*.

To unset **SQLRMDIR**, enter the following command:

```
unsetenv SQLRMDIR
```



Important: This environment variable functions differently in Version 5.0 and Version 6.0 and later Informix products. For details of Version 5.0 functionality, refer to Chapter 4 of the December 1991 release of this manual.



Tip: Version 5.0 or earlier clients connect to Informix database servers on Windows NT using INFORMIX-NET PC.

UNIX

TERM

The **TERM** environment variable is used for terminal handling. It enables DB-Access to recognize and communicate with the terminal that you are using.

```
setenv _____ TERM _____ type _____
```

type specifies the terminal type.

The terminal type specified in the **TERM** setting must correspond to an entry in the **termcap** file or **terminfo** directory. Before you can set the **TERM** environment variable, you must obtain the code for your terminal from the DBA.

For example, to specify the vt100 terminal, set the **TERM** environment variable by entering the following command:

```
setenv TERM vt100
```

UNIX

TERMCAP

The **TERMCAP** environment variable is used for terminal handling. It tells DB-Access to communicate with the **termcap** file instead of the **terminfo** directory.

```
setenv _____ TERMCAP _____ pathname _____
```

pathname specifies the location of the **termcap** file.

The **termcap** file contains a list of various types of terminals and their characteristics. For example, to provide DB-Access terminal-handling information, which is specified in the **/usr/informix/etc/termcap** file, enter the following command:

```
setenv TERMCAP /usr/informix/etc/termcap
```

You can use any of the following settings for **TERMCAP**. Use them in the following order:

1. The **termcap** file that you create
2. The **termcap** file that Informix supplies (that is, **\$INFORMIXDIR/etc/termcap**)
3. The operating-system **termcap** file (that is, **/etc/termcap**)

If you set the **TERMCAP** environment variable, be sure that the **INFORMIXTERM** environment variable is set to the default, **termcap**.

If you do not set the **TERMCAP** environment variable, the system file (that is, **/etc/termcap**) is used by default.

UNIX

TERMINFO

The **TERMINFO** environment variable is used for terminal handling.

The environment variable is supported only on platforms that provide full support for the **terminfo** libraries that System V and Solaris UNIX systems provide.

```
setenv _____ TERMINFO _____ /usr/lib/terminfo _____
```

TERMINFO tells DB-Access to communicate with the **terminfo** directory instead of the **termcap** file. The **terminfo** directory has subdirectories that contain files that pertain to terminals and their characteristics.

Enter the following command to set **TERMINFO**:

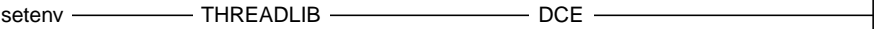
```
setenv TERMINFO /usr/lib/terminfo
```

If you set the **TERMINFO** environment variable, you must also set the **INFORMIXTERM** environment variable to **terminfo**.

THREADLIB

Use the **THREADLIB** environment variable to compile multithreaded ESQL/C applications. A multithreaded ESQL/C application lets you establish as many connections to one or more databases as there are threads. These connections can remain active while the application program executes.

The **THREADLIB** environment variable indicates which thread package to use when you compile an application. Currently only the Distributed Computing Environment (DCE) is supported.



The **THREADLIB** environment variable is checked when the **-thread** option is passed to the ESQL/C script when you compile a multithreaded ESQL/C application. When you use the **-thread** option while compiling, the ESQL/C script generates an error if the **THREADLIB** environment variable is not set or if the variable is set to an unsupported thread package.

Index of Environment Variables

Figure 3-1 provides an overview of the uses for the various Informix and UNIX environment variables that Version 7.3 and Version 8.2 support. It serves as an index to general topics and lists the related environment variables and the pages where the environment variables are introduced.

Figure 3-1
Environment Variables That Informix Products Use

Topic	Environment Variables	Page
ANSI compliance	DBANSIWARN	3-25
BYTE or TEXT data buffer	DBBLOBBUF	3-26
C compiler	INFORMIXC	3-50

(1 of 9)

Topic	Environment Variables	Page
C compiler: processing of multibyte characters	CC8BITLEVEL	Informix Guide to GLS Functionality
Client locale	CLIENT_LOCALE	Informix Guide to GLS Functionality
Client/server	INFORMIXSERVER	3-54
	INFORMIXSHMBASE	3-55
	INFORMIXSTACKSIZE	3-57
	SQLEXEC	3-67
	SQLRM	3-68
	SQLRMDIR	3-68
	CLIENT_LOCALE	Informix Guide to GLS Functionality
	DB_LOCALE	Informix Guide to GLS Functionality
	SERVER_LOCALE	Informix Guide to GLS Functionality
Code-set conversion	CLIENT_LOCALE	Informix Guide to GLS Functionality
	DB_LOCALE	Informix Guide to GLS Functionality
Communication Support Module: DCE-GSS	INFORMIXKEYTAB	3-53
Compilation: ESQL/C	THREADLIB	3-71

(2 of 9)

Topic	Environment Variables	Page
Compiler	CC8BITLEVEL	Informix Guide to GLS Functionality
	INFORMIXC	3-50
Configuration file: ignore variables	ENVIGNORE	3-48
Configuration file: ON-Archive	ARC_DEFAULT	3-22
Configuration file: Database server	ONCONFIG	3-60
Configuration file: tctermcap	ARC_KEYPAD	3-22
Connecting	INFORMIXCONRETRY	3-50
	INFORMIXCONTIME	3-51
	INFORMIXSERVER	3-54
	INFORMIXSQLHOSTS	3-56
Data distributions	DBUPSPACE	3-46
Database locale	DB_LOCALE	Informix Guide to GLS Functionality
Database server	INFORMIXSERVER	3-54
	SERVER_LOCALE	Informix Guide to GLS Functionality
Database server (obsolete)	SQLEXEC	3-67
Database server (obsolete)	SQLRM	3-68
	SQLRMDIR	3-68
Database server: archiving	ARC_DEFAULT	3-22
	ARC_KEYPAD	3-22
	DBREMOTECMD	3-40

Topic	Environment Variables	Page
Database server: configuration parameters	ONCONFIG	3-60
Database server: parallel sorting	PSORT_DBTEMP	3-64
	PSORT_NPROCS	3-65
Database server: role separation	INF_ROLE_SEP	3-59
Database server: shared memory	INFORMIXSHMBASE	3-55
Database server: stacksize	INFORMIXSTACKSIZE	3-57
Database server: tape management	ARC_DEFAULT	3-22
	ARC_KEYPAD	3-22
	DBREMOTECMD	3-40
Database server: temporary tables, sort files	DBSPACETEMP	3-41
Date and time values	DBCENTURY	3-26
	DBDATE	3-29, <i>Informix Guide to GLS Functionality</i>
	GL_DATE	Informix Guide to GLS Functionality
	GL_DATETIME	Informix Guide to GLS Functionality
	DBTIME	3-43
Delimited Identifiers	DELIMIDENT	3-46
Disk space	DBUPSPACE	3-46
Editor	DBEDIT	3-32
ESQL/C: C compiler	INFORMIXC	3-50

(4 of 9)

Topic	Environment Variables	Page
ESQL/C: running C preprocessor before the ESQL/C preprocessor	CPFIRST	3-23
ESQL/C: DATETIME formatting	DBTIME	3-43
ESQL/C: delimited identifiers	DELIMIDENT	3-46
ESQL/C: multibyte filter	ESQLMF	Informix Guide to GLS Functionality
ESQL/C: multibyte identifiers	CLIENT_LOCALE	Informix Guide to GLS Functionality
Executable programs	PATH	3-62
Fetch buffer size	FET_BUF_SIZE	3-48
Filenames: multibyte	GLS8BITSYS	Informix Guide to GLS Functionality
Files: field delimiter	DBDELIMITER	3-32
Files: installation	INFORMIXDIR	3-52
Files: locale	CLIENT_LOCALE	Informix Guide to GLS Functionality
	DB_LOCALE	Informix Guide to GLS Functionality
	SERVER_LOCALE	Informix Guide to GLS Functionality
Files: message	DBLANG	3-33
Files: temporary	DBSPACETEMP	3-41
Files: temporary sorting	PSORT_DBTEMP	3-64

(5 of 9)

Topic	Environment Variables	Page
Files: termcap , terminfo	INFORMIXTERM	3-58
	TERM	3-69
	TERMCAP	3-69
	TERMINFO	3-70
High-Performance Loader	DBONPLOAD	3-36
	PLCONFIG	3-64
Identifiers: delimited	DELIMIDENT	3-46
Identifiers: multibyte characters	CLIENT_LOCALE	Informix Guide to GLS Functionality
	ESQLMF	Informix Guide to GLS Functionality
Installation	INFORMIXDIR	3-52
	PATH	3-62
Language environment	DBLANG	3-33
Locale	CLIENT_LOCALE	Informix Guide to GLS Functionality
	DB_LOCALE	Informix Guide to GLS Functionality
	SERVER_LOCALE	Informix Guide to GLS Functionality
Message files	DBLANG	3-33
Money values	DBMONEY	3-35, Informix Guide to GLS Functionality

Topic	Environment Variables	Page
Multibyte characters	CLIENT_LOCALE	Informix Guide to GLS Functionality
	DB_LOCALE	Informix Guide to GLS Functionality
	SERVER_LOCALE	Informix Guide to GLS Functionality
Multibyte filter	ESQLMF	Informix Guide to GLS Functionality
Multithreaded applications	THREADLIB	3-71
Nondefault locale	CLIENT_LOCALE	Informix Guide to GLS Functionality
	DB_LOCALE	Informix Guide to GLS Functionality
	SERVER_LOCALE	Informix Guide to GLS Functionality
Optimization directives: setting in the query	IFX_DIRECTIVES	3-49
Pathname: for C COMPILER	INFORMIXC	3-50
Pathname: for database files	DBPATH	3-37
Pathname: for executable programs	PATH	3-62
Pathname: for installation	INFORMIXDIR	3-52
Pathname: for message files	DBLANG	3-33
Pathname: for parallel sorting	PSORT_DBTEMP	3-64

(7 of 9)

Topic	Environment Variables	Page
Pathname: for relay module	SQLEXEC	3-67
Pathname: for remote shell	DBREMOTECMD	3-40
Printing	DBPRINT	3-39
Privileges	NODEFDAC	3-59
Program: printing	DBPRINT	3-39
Relay module	SQLEXEC	3-67
	SQLRM	3-68
	SQLRMDIR	3-68
Remote shell	DBREMOTECMD	3-40
Role separation	INF_ROLE_SEP	3-59
Routine: DATETIME formatting	DBTIME	3-43
Server	See <i>Database server</i>.	
Server locale	SERVER_LOCALE	Informix Guide to GLS Functionality
Shared memory	INFORMIXSHMBASE	3-55
Shell: remote	DBREMOTECMD	3-40
Shell: search path	PATH	3-62
Sorting	PSORT_DBTEMP	3-64
	PSORT_NPROCS	3-65
	DBSPACETEMP	3-41
SQL statement: CONNECT	INFORMIXSERVER	3-54
SQL statement: editing	DBEDIT	3-32
SQL statement: LOAD, UNLOAD	DBDELIMITER	3-32
SQL statement: UPDATE STATISTICS	DBUPSPACE	3-46

Topic	Environment Variables	Page
Stacksize	INFORMIXSTACKSIZE	3-57
Tables: temporary	DBSPACETEMP	3-41
	PSORT_DBTEMP	3-64
Temporary tables	DBSPACETEMP	3-41
Terminal handling	INFORMIXTERM	3-58
	TERM	3-69
	TERMCAP	3-69
	TERMINFO	3-70
Utilities: DB-Access	DBDELIMITER	3-32
	DBEDIT	3-32
	INFORMIXTERM	3-58
	DBFLTMASK	3-33
Utilities: dbexport	DBDELIMITER	3-32
Utilities: ON-Archive	ARC_DEFAULT	3-22
	ARC_KEYPAD	3-22
	DBREMOTECMD	3-40
Values: date and time	DBDATE	3-29, <i>Informix Guide to GLS Functionality</i>
	DBTIME	3-43
Values: money	DBMONEY	3-35
Variables: overriding	ENVIGNORE	3-48

The stores7 Database

A

The **stores7** database contains a set of tables that describe an imaginary business. The examples in the [Informix Guide to SQL: Syntax](#) and [Informix Guide to SQL: Tutorial](#) are based on this database. The **stores7** database is not ANSI compliant.

This appendix contains the following sections:

- The first section describes the structure of the tables in the **stores7** database. It identifies the primary key of each table, lists the name and data type of each column, and indicates whether the column has a default value or check constraint. Indexes on columns are also identified and classified as unique or if they allow duplicate values.
- The second section shows a graphic map of the tables in the **stores7** database and indicates the relationships between columns.
- The third section describes the primary-foreign key relationships between columns in tables.
- The final section shows the data contained in each table of the **stores7** database.

Structure of the Tables

The **stores7** database contains information about a fictitious sporting-goods distributor that services stores in the Western United States. This database includes the following tables:

- **customer**
- **orders**
- **items**
- **stock**
- **catalog**
- **cust_calls**
- **call_type**
- **manufact**
- **state**


The following sections describe each table. The unique identifier for each table (primary key) is shaded and indicated by a key symbol.

The customer Table

The **customer** table contains information about the retail stores that place orders from the distributor. Figure A-1 shows the columns of the **customer** table.

The **zipcode** column in Figure A-1 is indexed and allows duplicate values.


Figure A-1
The customer Table

Column Name	Data Type	Description
 customer_num	SERIAL(101)	system-generated customer number
fname	CHAR(15)	first name of store representative
lname	CHAR(15)	last name of store representative
company	CHAR(20)	name of store
address1	CHAR(20)	first line of store address
address2	CHAR(20)	second line of store address
city	CHAR(15)	city
state	CHAR(18)	state (foreign key to state table)
zipcode	CHAR(2)	zipcode
phone	CHAR(5)	telephone number

The orders Table

The **orders** table contains information about orders placed by the customers of the distributor. Figure A-2 shows the columns of the **orders** table.


Figure A-2
The orders Table

Column Name	Data Type	Description
 order_num	SERIAL(1001)	system-generated order number
order_date	DATE	date order entered
customer_num	INTEGER	customer number (foreign key to customer table)
ship_instruct	CHAR(40)	special shipping instructions
backlog	CHAR(1)	indicates order cannot be filled because the item is backlogged: y = yes n = no
po_num	CHAR(10)	customer purchase order number
ship_date	DATE	shipping date
ship_weight	DECIMAL(8,2)	shipping weight
ship_charge	MONEY(6)	shipping charge
paid_date	DATE	date order paid

The items Table

An order can include one or more items. One row exists in the **items** table for each item in an order. Figure A-3 shows the columns of the **items** table.

Figure A-3
The items Table


Column Name	Data Type	Description
 item_num	SMALLINT	sequentially assigned item number for an order
order_num	INTEGER	order number (foreign key to orders table)
stock_num	SMALLINT	stock number for item (foreign key to stock table)
manu_code	CHAR(3)	manufacturer code for item ordered (foreign key to manufact table)
quantity	SMALLINT	quantity ordered (value must be > 1)
total_price	MONEY(8)	quantity ordered * unit price = total price of item

The stock Table

The distributor carries 41 types of sporting goods from various manufacturers. More than one manufacturer can supply an item. For example, the distributor offers racer goggles from two manufacturers and running shoes from six manufacturers.

The **stock** table is a catalog of the items sold by the distributor. Figure A-4 shows the columns of the **stock** table.


Figure A-4
The stock Table

Column Name	Data Type	Description
 stock_num	SMALLINT	stock number that identifies type of item
manu_code	CHAR(3)	manufacturer code (foreign key to manufact table)
description	CHAR(15)	description of item
unit_price	MONEY(6,2)	unit price
unit	CHAR(4)	unit by which item is ordered: each pair case box
unit_descr	CHAR(15)	description of unit

The catalog Table

The **catalog** table describes each item in stock. Retail stores use this table when placing orders with the distributor. Figure A-5 shows the columns of the **catalog** table.

Figure A-5
The catalog Table

Column Name	Data Type	Description
 catalog_num	SERIAL(10001)	system-generated catalog number
stock_num	SMALLINT	distributor stock number (foreign key to stock table)
manu_code	CHAR(3)	manufacturer code (foreign key to manufact table)
cat_descr	TEXT	description of item
cat_picture	BYTE	picture of item (binary data)
cat_advert	VARCHAR(255, 65)	tag line underneath picture

The cust_calls Table

All customer calls for information on orders, shipments, or complaints are logged. The **cust_calls** table contains information about these types of customer calls. Figure A-6 shows the columns of the **cust_calls** table.


Column Name	Data Type	Description
 customer_num	INTEGER	customer number (foreign key to customer table)
call_dtime	DATETIME YEAR TO MINUTE	date and time call received
user_id	CHAR(18)	name of person logging call (default is user login name)
call_code	CHAR(1)	type of call (foreign key to call_type table)
call_descr	CHAR(240)	description of call
res_dtime	DATETIME YEAR TO MINUTE	date and time call resolved
res_descr	CHAR(240)	description of how call was resolved

Figure A-6
The cust_calls Table

The call_type Table

The call codes associated with customer calls are stored in the **call_type** table. Figure A-7 shows the columns of the **call_type** table.


Column Name	Data Type	Description
 call_code	CHAR(1)	call code
call_descr	CHAR (30)	description of call type

Figure A-7
The call_type Table

The manufact Table

Information about the nine manufacturers whose sporting goods are handled by the distributor is stored in the **manufact** table. Figure A-8 shows the columns of the **manufact** table.


Column Name	Data Type	Description
 manu_code	CHAR(3)	manufacturer code
manu_name	CHAR(15)	name of manufacturer
lead_time	INTERVAL DAY(3) TO DAY	lead time for shipment of orders

Figure A-8
The manufact Table

The state Table

The **state** table contains the names and postal abbreviations for the 50 states of the United States. Figure A-9 shows the columns of the **state** table.


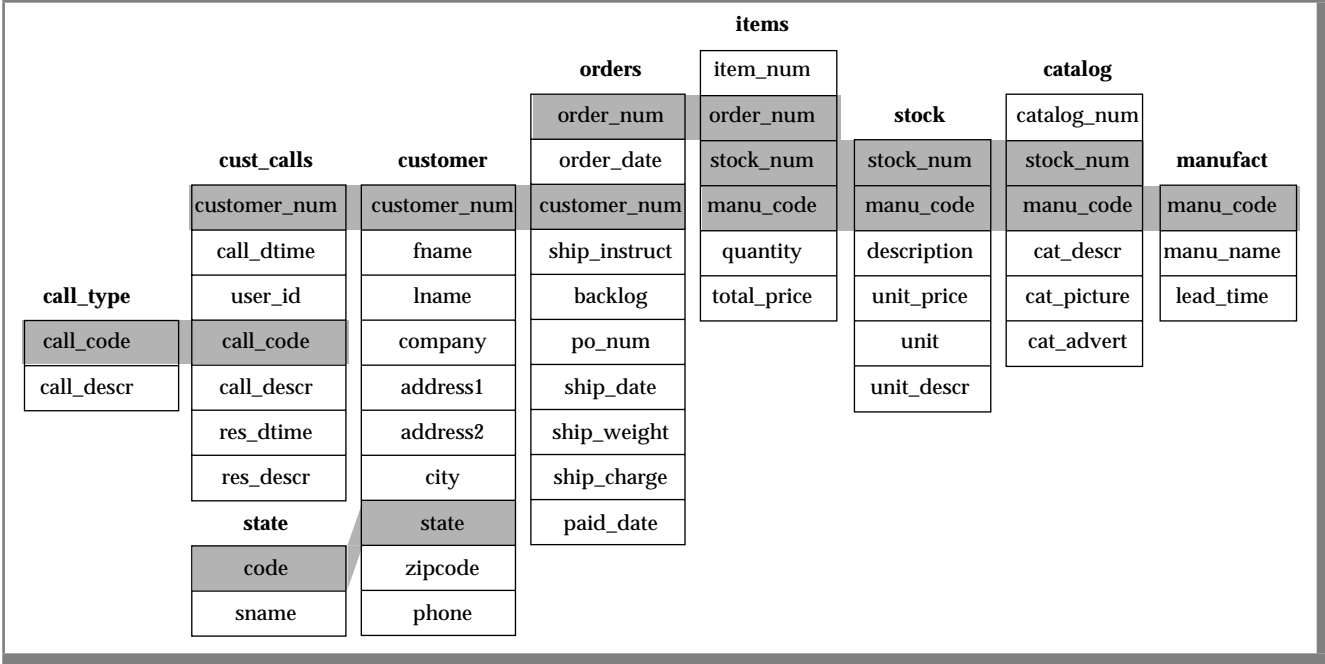
Name	Type	Description
 code	CHAR(2)	state code
sname	CHAR(15)	state name

Figure A-9
The state Table

The stores7 Database Map

Figure A-10 displays the joins in the **stores7** database. The grey shading that connects a column in one table to the same column in another table indicates the relationships, or *joins*, between tables.

Figure A-10
Joins in the stores7 Database



Primary-Foreign Key Relationships

The tables of the **stores7** database are linked by the primary-foreign key relationships that Figure A-10 shows and are identified in this section. This type of relationship is called a *referential constraint* because a foreign key in one table *references* the primary key in another table. Figure A-11 through Figure A-18 show the relationships among tables and how information stored in one table supplements information stored in others.

The customer and orders Tables

The **customer** table contains a **customer_num** column that holds a number identifying a customer, along with columns for the customer name, company, address, and telephone number. For example, the row with information about Anthony Higgins contains the number 104 in the **customer_num** column. The **orders** table also contains a **customer_num** column that stores the number of the customer who placed a particular order. In the **orders** table, the **customer_num** column is a foreign key that references the **customer_num** column in the **customer** table. Figure A-11 shows this relationship.

customer Table (detail)

customer_num	fname	lname
101	Ludwig	Pauli
102	Carole	Sadler
103	Philip	Currie
104	Anthony	Higgins

orders Table (detail)

order_num	order_date	customer_num
1001	05/20/1998	104
1002	05/21/1998	101
1003	05/22/1998	104
1004	05/22/1998	106

Figure A-11
Tables That the
customer_num
Column Joins

According to Figure A-11, customer 104 (Anthony Higgins) has placed two orders, as his customer number appears in two rows of the **orders** table. Because the customer number is a foreign key in the **orders** table, you can retrieve Anthony Higgins' name, address, and information about his orders at the same time.

The orders and items Tables

The **orders** and **items** tables are linked by an **order_num** column that contains an identification number for each order. If an order includes several items, the same order number appears in several rows of the **items** table. In the **items** table, the **order_num** column is a foreign key that references the **order_num** column in the **orders** table. Figure A-12 shows this relationship.

orders Table (detail)			
order_num	order_date	customer_num	
1001	05/20/1998	104	
1002	05/21/1998	101	
1003	05/22/1998	104	

items Table (detail)			
item_num	order_num	stock_num	manu_code
1	1001	1	HRO
1	1002	4	HSK
2	1002	3	HSK
1	1003	9	ANZ
2	1003	8	ANZ
3	1003	5	ANZ

Figure A-12
Tables That the
order_num Column
Joins

The items and stock Tables

The **items** table and the **stock** table are joined by two columns: the **stock_num** column, which stores a stock number for an item, and the **manu_code** column, which stores a code that identifies the manufacturer. You need both the stock number and the manufacturer code to uniquely identify an item. For example, the item with the stock number 1 and the manufacturer code **HRO** is a Hero baseball glove; the item with the stock number 1 and the manufacturer code **HSK** is a Husky baseball glove. The same stock number and manufacturer code can appear in more than one row of the **items** table, if the same item belongs to separate orders. In the **items** table, the **stock_num** and **manu_code** columns are foreign keys that reference the **stock_num** and **manu_code** columns in the **stock** table. Figure A-13 shows this relationship.

items Table (detail)			
item_num	order_num	stock_num	manu_code
1	1001	1	HRO
1	1002	4	HSK
2	1002	3	HSK
1	1003	9	ANZ
2	1003	8	ANZ
3	1003	5	ANZ
1	1004	1	HRO

stock Table (detail)		
stock_num	manu_code	description
1	HRO	baseball gloves
1	HSK	baseball gloves
1	SMT	baseball gloves

Figure A-13
Tables That the
stock_num and
manu_code
Columns Join

The stock and catalog Tables

The **stock** table and **catalog** table are joined by two columns: the **stock_num** column, which stores a stock number for an item, and the **manu_code** column, which stores a code that identifies the manufacturer. You need both columns to uniquely identify an item. In the **catalog** table, the **stock_num** and **manu_code** columns are foreign keys that reference the **stock_num** and **manu_code** columns in the **stock** table. Figure A-14 shows this relationship.

Figure A-14
*Tables That the
stock_num and
manu_code
Columns Join*

stock Table (detail)		
stock_num	manu_code	description
1	HRO	baseball gloves
1	HSK	baseball gloves
1	SMT	baseball gloves

catalog Table (detail)		
catalog_num	stock_num	manu_code
10001	1	HRO
10002	1	HSK
10003	1	SMT
10004	2	HRO

The stock and manufact Tables

The **stock** table and the **manufact** table are joined by the **manu_code** column. The same manufacturer code can appear in more than one row of the **stock** table if the manufacturer produces more than one piece of equipment. In the **stock** table, the **manu_code** column is a foreign key that references the **manu_code** column in the **manufact** table. Figure A-15 shows this relationship.

stock Table (detail)		
stock_num	manu_code	description
1	HRO	baseball gloves
1	HSK	baseball gloves
1	SMT	baseball gloves
manufact Table (detail)		
manu_code	manu_name	
NRG	Norge	
HSK	Husky	
HRO	Hero	

Figure A-15
Tables That the
manu_code
Column Joins

The cust_calls and customer Tables

The **cust_calls** table and the **customer** table are joined by the **customer_num** column. The same customer number can appear in more than one row of the **cust_calls** table if the customer calls the distributor more than once with a problem or question. In the **cust_calls** table, the **customer_num** column is a foreign key that references the **customer_num** column in the **customer** table. Figure A-16 shows this relationship.

Figure A-16
Tables That the
customer_num
Column Joins

customer Table (detail)		
customer_num	fname	lname
101	Ludwig	Pauli
102	Carole	Sadler
103	Philip	Currie
104	Anthony	Higgins
105	Raymond	Vector
106	George	Watson

cust_calls Table (detail)		
customer_num	call_dtime	user_id
106	1998-06-12 08:20	maryj
127	1998-07-31 14:30	maryj
116	1997-11-28 13:34	mannyh
116	1997-12-21 11:24	mannyh

The call_type and cust_calls Table

The **call_type** and **cust_calls** tables are joined by the **call_code** column. The same call code can appear in more than one row of the **cust_calls** table because many customers can have the same *type* of problem. In the **cust_calls** table, the **call_code** column is a foreign key that references the **call_code** column in the **call_type** table. Figure A-17 shows this relationship.

call_type Table (detail)

call_code	code_descr
B	billing error
D	damaged goods
I	incorrect merchandise sent
L	late shipment
O	other

cust_calls Table (detail)

customer_num	call_dtime	call_code
106	1998-06-12 08:20	D
127	1998-07-31 14:30	I
116	1997-11-28 13:34	I
116	1997-12-21 11:24	I

Figure A-17
Tables That the
call_code Column
Joins

The state and customer Tables

The **state** table and the **customer** table are joined by a column that contains the state code. This column is called **code** in the **state** table and **state** in the **customer** table. If several customers live in the same state, the same state code appears in several rows of the table. In the **customer** table, the **state** column is a foreign key that references the **code** column in the **state** table. Figure A-18 shows this relationship.

customer Table (detail)				
customer_num	fname	lname	---	state
101	Ludwig	Pauli	---	CA
102	Carole	Sadler	---	CA
103	Philip	Currie	---	CA
state Table (detail)				
code		sname		
AK		Alaska		
AL		Alabama		
AR		Arkansas		
AZ		Arizona		
CA		California		

Figure A-18
Tables That the
state/code Column
Joins

Data in the stores7 Database

The following tables display the data in the **stores7** database.

customer Table

customer_num	fname	lname	company	address1	address2	city	state	zipcode	phone
101	Ludwig	Pauli	All Sports Supplies	213 Erstwild Court		Sunnyvale	CA	94086	408-789-8075
102	Carole	Sadler	Sports Spot	785 Geary St		San Francisco	CA	94117	415-822-1289
103	Philip	Currie	Phil's Sports	654 Poplar	P. O. Box 3498	Palo Alto	CA	94303	650-328-4543
104	Anthony	Higgins	Play Ball!	East Shopping Cntr.	422 Bay Road	Redwood City	CA	94026	650-368-1100
105	Raymond	Vector	Los Altos Sports	1899 La Loma Drive		Los Altos	CA	94022	650-776-3249
106	George	Watson	Watson & Son	1143 Carver Place		Mountain View	CA	94063	650-389-8789
107	Charles	Ream	Athletic Supplies	41 Jordan Avenue		Palo Alto	CA	94304	650-356-9876
108	Donald	Quinn	Quinn's Sports	587 Alvarado		Redwood City	CA	94063	650-544-8729
109	Jane	Miller	Sport Stuff	Mayfair Mart	7345 Ross Blvd.	Sunnyvale	CA	94086	408-723-8789
110	Roy	Jaeger	AA Athletics	520 Topaz Way		Redwood City	CA	94062	650-743-3611
111	Frances	Keyes	Sports Center	3199 Sterling Court		Sunnyvale	CA	94085	408-277-7245
112	Margaret	Lawson	Runners & Others	234 Wyandotte Way		Los Altos	CA	94022	650-887-7235
113	Lana	Beatty	Sportstown	654 Oak Grove		Menlo Park	CA	94025	650-356-9982
114	Frank	Albertson	Sporting Place	947 Waverly Place		Redwood City	CA	94062	650-886-6677
115	Alfred	Grant	Gold Medal Sports	776 Gary Avenue		Menlo Park	CA	94025	650-356-1123

customer_num	fname	lname	company	address1	address2	city	state	zipcode	phone
116	Jean	Parmelee	Olympic City	1104 Spinosa Drive		Mountain View	CA	94040	650-534-8822
117	Arnold	Sipes	Kids Korner	850 Lytton Court		Redwood City	CA	94063	650-245-4578
118	Dick	Baxter	Blue Ribbon Sports	5427 College		Oakland	CA	94609	650-655-0011
119	Bob	Shorter	The Triathletes Club	2405 Kings Highway		Cherry Hill	NJ	08002	609-663-6079
120	Fred	Jewell	Century Pro Shop	6627 N. 17th Way		Phoenix	AZ	85016	602-265-8754
121	Jason	Wallack	City Sports	Lake Biltmore Mall	350 W. 23rd Street	Wilmington	DE	19898	302-366-7511
122	Cathy	O'Brian	The Sporting Life	543 Nassau Street		Princeton	NJ	08540	609-342-0054
123	Marvin	Hanlon	Bay Sports	10100 Bay Meadows Rd	Suite 1020	Jacksonville	FL	32256	904-823-4239
124	Chris	Putnum	Putnum's Putters	4715 S.E. Adams Blvd	Suite 909C	Bartlesville	OK	74006	918-355-2074
125	James	Henry	Total Fitness Sports	1450 Commonwealth Ave.		Brighton	MA	02135	617-232-4159

customer_num	fname	lname	company	address1	address2	city	state	zipcode	phone
126	Eileen	Neelie	Neelie's Discount Sports	2539 South Utica St		Denver	CO	80219	303-936-7731
127	Kim	Satifer	Big Blue Bike Shop	Blue Island Square	12222 Gregory Street	Blue Island	NY	60406	312-944-5691
128	Frank	Lessor	Phoenix University	Athletic Department	1817 N. Thomas Road	Phoenix	AZ	85008	602-533-1817

(3 of 3)

Data in the stores7 Database

Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech

items Table

item_num	order_num	stock_num	manu_code	quantity	total_price
1	1001	1	HRO	1	250.00
1	1002	4	HSK	1	960.00
2	1002	3	HSK	1	240.00
1	1003	9	ANZ	1	20.00
2	1003	8	ANZ	1	840.00
3	1003	5	ANZ	5	99.00
1	1004	1	HRO	1	250.00
2	1004	2	HRO	1	126.00
3	1004	3	HSK	1	240.00
4	1004	1	HSK	1	800.00
1	1005	5	NRG	10	280.00
2	1005	5	ANZ	10	198.00
3	1005	6	SMT	1	36.00
4	1005	6	ANZ	1	48.00
1	1006	5	SMT	5	125.00
2	1006	5	NRG	5	140.00
3	1006	5	ANZ	5	99.00
4	1006	6	SMT	1	36.00
5	1006	6	ANZ	1	48.00
1	1007	1	HRO	1	250.00
2	1007	2	HRO	1	126.00
3	1007	3	HSK	1	240.00
4	1007	4	HRO	1	480.00

(1 of 3)

item_num	order_num	stock_num	manu_code	quantity	total_price
5	1007	7	HRO	1	600.00
1	1008	8	ANZ	1	840.00
2	1008	9	ANZ	5	100.00
1	1009	1	SMT	1	450.00
1	1010	6	SMT	1	36.00
2	1010	6	ANZ	1	48.00
1	1011	5	ANZ	5	99.00
1	1012	8	ANZ	1	840.00
2	1012	9	ANZ	10	200.00
1	1013	5	ANZ	1	19.80
2	1013	6	SMT	1	36.00
3	1013	6	ANZ	1	48.00
4	1013	9	ANZ	2	40.00
1	1014	4	HSK	1	960.00
2	1014	4	HRO	1	480.00
1	1015	1	SMT	1	450.00
1	1016	101	SHM	2	136.00
2	1016	109	PRC	3	90.00
3	1016	110	HSK	1	308.00
4	1016	114	PRC	1	120.00
1	1017	201	NKL	4	150.00
2	1017	202	KAR	1	230.00
3	1017	301	SHM	2	204.00
1	1018	307	PRC	2	500.00

(2 of 3)

Data in the stores7 Database

Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech

item_num	order_num	stock_num	manu_code	quantity	total_price
2	1018	302	KAR	3	15.00
3	1018	110	PRC	1	236.00
4	1018	5	SMT	4	100.00
5	1018	304	HRO	1	280.00
1	1019	111	SHM	3	1499.97
1	1020	204	KAR	2	90.00
2	1020	301	KAR	4	348.00
1	1021	201	NKL	2	75.00
2	1021	201	ANZ	3	225.00
3	1021	202	KAR	3	690.00
4	1021	205	ANZ	2	624.00
1	1022	309	HRO	1	40.00
2	1022	303	PRC	2	96.00
3	1022	6	ANZ	2	96.00
1	1023	103	PRC	2	40.00
2	1023	104	PRC	2	116.00
3	1023	105	SHM	1	80.00
4	1023	110	SHM	1	228.00
5	1023	304	ANZ	1	170.00
6	1023	306	SHM	1	190.00

(3 of 3)

call_type Table

call_code	code_descr
B	billing error
D	damaged goods
I	incorrect merchandise sent
L	late shipment
O	other

orders Table

order_num	order_date	customer_num	ship_instruct	backlog	po_num	ship_date	ship_weight	ship_charge	paid_date
1001	05/20/1998	104	express	n	B77836	06/01/1998	20.40	10.00	07/22/1998
1002	05/21/1998	101	PO on box; deliver back door only	n	9270	05/26/1998	50.60	15.30	06/03/1998
1003	05/22/1998	104	express	n	B77890	05/23/1998	35.60	10.80	06/14/1998
1004	05/22/1998	106	ring bell twice	y	8006	05/30/1998	95.80	19.20	
1005	05/24/1998	116	call before delivery	n	2865	06/09/1998	80.80	16.20	06/21/1998
1006	05/30/1998	112	after 10AM	y	Q13557		70.80	14.20	
1007	05/31/1998	117		n	278693	06/05/1998	125.90	25.20	
1008	06/07/1998	110	closed Monday	y	LZ230	07/06/1998	45.60	13.80	07/21/1998
1009	06/14/1998	111	door next to grocery	n	4745	06/21/1998	20.40	10.00	08/21/1998
1010	06/17/1998	115	deliver 776 King St. if no answer	n	429Q	06/29/1998	40.60	12.30	08/22/1998
1011	06/18/1998	104	express	n	B77897	07/03/1998	10.40	5.00	08/29/1998
1012	06/18/1998	117		n	278701	06/29/1998	70.80	14.20	

order_num	order_date	customer_num	ship_instruct	backlog	po_num	ship_date	ship_weight	ship_charge	paid_date
1013	06/22/1998	104	express	n	B77930	07/10/1998	60.80	12.20	07/31/1998
1014	06/25/1998	106	ring bell, kick door loudly	n	8052	07/03/1998	40.60	12.30	07/10/1998
1015	06/27/1998	110	closed Mondays	n	MA003	07/16/1998	20.60	6.30	08/31/1998
1016	06/29/1998	119	delivery entrance off Camp St.	n	PC6782	07/12/1998	35.00	11.80	
1017	07/09/1998	120	North side of clubhouse	n	DM3543 31	07/13/1998	60.00	18.00	
1018	07/10/1998	121	SW corner of Biltmore Mall	n	S22942	07/13/1998	70.50	20.00	08/06/1998
1019	07/11/1998	122	closed til noon Mondays	n	Z55709	07/16/1998	90.00	23.00	08/06/1998
1020	07/11/1998	123	express	n	W2286	07/16/1998	14.00	8.50	09/20/1998
1021	07/23/1998	124	ask for Elaine	n	C3288	07/25/1998	40.00	12.00	08/22/1998
1022	07/24/1998	126	express	n	W9925	07/30/1998	15.00	13.00	09/02/1998
1023	07/24/1998	127	no deliveries after 3 p.m.	n	KF2961	07/30/1998	60.00	18.00	08/22/1998

(2 of 2)

stock Table

stock_num	manu_code	description	unit_price	unit	unit_descr
1	HRO	baseball gloves	250.00	case	10 gloves/case
1	HSK	baseball gloves	800.00	case	10 gloves/case
1	SMT	baseball gloves	450.00	case	10 gloves/case
2	HRO	baseball	126.00	case	24/case
3	HSK	baseball bat	240.00	case	12/case
3	SHM	baseball bat	280.00	case	12/case
4	HSK	football	960.00	case	24/case
4	HRO	football	480.00	case	24/case
5	NRG	tennis racquet	28.00	each	each
5	SMT	tennis racquet	25.00	each	each
5	ANZ	tennis racquet	19.80	each	each
6	SMT	tennis ball	36.00	case	24 cans/case
6	ANZ	tennis ball	48.00	case	24 cans/case
7	HRO	basketball	600.00	case	24/case
8	ANZ	volleyball	840.00	case	24/case
9	ANZ	volleyball net	20.00	each	each
101	PRC	bicycle tires	88.00	box	4/box
101	SHM	bicycle tires	68.00	box	4/box
102	SHM	bicycle brakes	220.00	case	4 sets/case
102	PRC	bicycle brakes	480.00	case	4 sets/case
103	PRC	front derailleur	20.00	each	each
104	PRC	rear derailleur	58.00	each	each
105	PRC	bicycle wheels	53.00	pair	pair

(1 of 4)

stock_num	manu_code	description	unit_price	unit	unit_descr
105	SHM	bicycle wheels	80.00	pair	pair
106	PRC	bicycle stem	23.00	each	each
107	PRC	bicycle saddle	70.00	pair	pair
108	SHM	crankset	45.00	each	each
109	PRC	pedal binding	30.00	case	6 pairs/case
109	SHM	pedal binding	200.00	case	4 pairs/case
110	PRC	helmet	236.00	case	4/case
110	ANZ	helmet	244.00	case	4/case
110	SHM	helmet	228.00	case	4/case
110	HRO	helmet	260.00	case	4/case
110	HSK	helmet	308.00	case	4/case
111	SHM	10-spd, assmbld	499.99	each	each
112	SHM	12-spd, assmbld	549.00	each	each
113	SHM	18-spd, assmbld	685.90	each	each
114	PRC	bicycle gloves	120.00	case	10 pairs/case
201	NKL	golf shoes	37.50	each	each
201	ANZ	golf shoes	75.00	each	each
201	KAR	golf shoes	90.00	each	each
202	NKL	metal woods	174.00	case	2 sets/case
202	KAR	std woods	230.00	case	2 sets/case
203	NKL	irons/wedges	670.00	case	2 sets/case
204	KAR	putter	45.00	each	each
205	NKL	3 golf balls	312.00	case	24/case
205	ANZ	3 golf balls	312.00	case	24/case

(2 of 4)

Data in the stores7 Database

Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech Review Draft Tech

stock_num	manu_code	description	unit_price	unit	unit_descr
205	HRO	3 golf balls	312.00	case	24/case
301	NKL	running shoes	97.00	each	each
301	HRO	running shoes	42.50	each	each
301	SHM	running shoes	102.00	each	each
301	PRC	running shoes	75.00	each	each
301	KAR	running shoes	87.00	each	each
301	ANZ	running shoes	95.00	each	each
302	HRO	ice pack	4.50	each	each
302	KAR	ice pack	5.00	each	each
303	PRC	socks	48.00	box	24 pairs/box
303	KAR	socks	36.00	box	24 pair/box
304	ANZ	watch	170.00	box	10/box
304	HRO	watch	280.00	box	10/box
305	HRO	first-aid kit	48.00	case	4/case
306	PRC	tandem adapter	160.00	each	each
306	SHM	tandem adapter	190.00	each	each
307	PRC	infant jogger	250.00	each	each
308	PRC	twin jogger	280.00	each	each
309	HRO	ear drops	40.00	case	20/case
309	SHM	ear drops	40.00	case	20/case
310	SHM	kick board	80.00	case	10/case
310	ANZ	kick board	89.00	case	12/case
311	SHM	water gloves	48.00	box	4 pairs/box
312	SHM	racer goggles	96.00	box	12/box

(3 of 4)

stock_num	manu_code	description	unit_price	unit	unit_descr
312	HRO	racer goggles	72.00	box	12/box
313	SHM	swim cap	72.00	box	12/box
313	ANZ	swim cap	60.00	box	12/box

(4 of 4)

catalog Table

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10001	1	HRO	Brown leather. Specify first baseman's or infield/outfield style. Specify right- or left-handed.	<BYTE value>	Your First Season's Baseball Glove
10002	1	HSK	Babe Ruth signature glove. Black leather. Infield/outfield style. Specify right- or left-handed.	<BYTE value>	All-Leather, Hand-Stitched, Deep-Pockets, Sturdy Webbing that Won't Let Go
10003	1	SMT	Catcher's mitt. Brown leather. Specify right- or left-handed.	<BYTE value>	A Sturdy Catcher's Mitt With the Perfect Pocket
10004	2	HRO	Jackie Robinson signature glove. Highest Professional quality, used by National League.	<BYTE value>	Highest Quality Ball Available, from the Hand-Stitching to the Robinson Signature
10005	3	HSK	Pro-style wood. Available in sizes: 31, 32, 33, 34, 35.	<BYTE value>	High-Technology Design Expands the Sweet Spot
10006	3	SHM	Aluminum. Blue with black tape. 31", 20 oz or 22 oz; 32", 21 oz or 23 oz; 33", 22 oz or 24 oz.	<BYTE value>	Durable Aluminum for High School and Collegiate Athletes
10007	4	HSK	Norm Van Brocklin signature style.	<BYTE value>	Quality Pigskin with Norm Van Brocklin Signature
10008	4	HRO	NFL-Style pigskin.	<BYTE value>	Highest Quality Football for High School and Collegiate Competitions

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10009	5	NRG	Graphite frame. Synthetic strings.	<BYTE value>	Wide Body Amplifies Your Natural Abilities by Providing More Power Through Aerodynamic Design
10010	5	SMT	Aluminum frame. Synthetic strings.	<BYTE value>	Mid-Sized Racquet for the Improving Player
10011	5	ANZ	Wood frame, cat-gut strings.	<BYTE value>	Antique Replica of Classic Wooden Racquet Built with Cat-Gut Strings
10012	6	SMT	Soft yellow color for easy visibility in sunlight or artificial light.	<BYTE value>	High-Visibility Tennis, Day or Night
10013	6	ANZ	Pro-core. Available in neon yellow, green, and pink.	<BYTE value>	Durable Construction Coupled with the Brightest Colors Available
10014	7	HRO	Indoor. Classic NBA style. Brown leather.	<BYTE value>	Long-Life Basketballs for Indoor Gymnasiums
10015	8	ANZ	Indoor. Finest leather. Professional quality.	<BYTE value>	Professional Volleyballs for Indoor Competitions
10016	9	ANZ	Steel eyelets. Nylon cording. Double-stitched. Sanctioned by the National Athletic Congress.	<BYTE value>	Sanctioned Volleyball Netting for Indoor Professional and Collegiate Competition
10017	101	PRC	Reinforced, hand-finished tubular. Polyurethane belted. Effective against punctures. Mixed tread for super wear and road grip.	<BYTE value>	Ultimate in Puncture Protection, Tires Designed for In-City Riding

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10018	101	SHM	Durable nylon casing with butyl tube for superior air retention. Center-ribbed tread with herringbone side. Coated sidewalls resist abrasion.	<BYTE value>	The Perfect Tire for Club Rides or Training
10019	102	SHM	Thrust bearing and coated pivot washer/ spring sleeve for smooth action. Slotted levers with soft gum hoods. Two-tone paint treatment. Set includes calipers, levers, and cables.	<BYTE value>	Thrust-Bearing and Spring-Sleeve Brake Set Guarantees Smooth Action
10020	102	PRC	Computer-aided design with low-profile pads. Cold-forged alloy calipers and beefy caliper bushing. Aero levers. Set includes calipers, levers, and cables.	<BYTE value>	Computer Design Delivers Rigid Yet Vibration-Free Brakes
10021	103	PRC	Compact leading-action design enhances shifting. Deep cage for super-small granny gears. Extra strong construction to resist off-road abuse.	<BYTE value>	Climb Any Mountain: ProCycle's Front Derailleur Adds Finesse to Your ATB
10022	104	PRC	Floating trapezoid geometry with extra thick parallelogram arms. 100-tooth capacity. Optimum alignment with any freewheel.	<BYTE value>	Computer-Aided Design Engineers 100-Tooth Capacity Into ProCycle's Rear Derailleur
10023	105	PRC	Front wheels laced with 15g spokes in a 3-cross pattern. Rear wheels laced with 14g spikes in a 3-cross pattern.	<BYTE value>	Durable Training Wheels That Hold True Under Toughest Conditions

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10024	105	SHM	Polished alloy. Sealed-bearing, quick-release hubs. Double-buttet. Front wheels are laced 15g/2-cross. Rear wheels are laced 15g/3-cross.	<BYTE value>	Extra Lightweight Wheels for Training or High-Performance Touring
10025	106	PRC	Hard anodized alloy with pearl finish. 6mm hex bolt hardware. Available in lengths of 90-140mm in 10mm increments.	<BYTE value>	ProCycle Stem with Pearl Finish
10026	107	PRC	Available in three styles: Men's racing; Men's touring; and Women's. Anatomical gel construction with lycra cover. Black or black/hot pink.	<BYTE value>	The Ultimate In Riding Comfort, Lightweight With Anatomical Support
10027	108	SHM	Double or triple crankset with choice of chainrings. For double crankset, chainrings from 38-54 teeth. For triple crankset, chainrings from 24-48 teeth.	<BYTE value>	Customize Your Mountain Bike With Extra-Durable Crankset
10028	109	PRC	Steel toe clips with nylon strap. Extra wide at buckle to reduce pressure.	<BYTE value>	Classic Toeclip Improved To Prevent Soreness At Clip Buckle

(4 of 12)

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10029	109	SHM	Ingenious new design combines button on sole of shoe with slot on a pedal plate to give riders new options in riding efficiency. Choose full or partial locking. Four plates mean both top and bottom of pedals are slotted—no fishing around when you want to engage full power. Fast unlocking ensures safety when maneuverability is paramount.	<BYTE value>	Ingenious Pedal/Clip Design Delivers Maximum Power And Fast Unlocking
10030	110	PRC	Super-lightweight. Meets both ANSI and Snell standards for impact protection. 7.5 oz. Quick-release shadow buckle.	<BYTE value>	Feather-Light, Quick-Release, Maximum Protection Helmet
10031	110	ANZ	No buckle so no plastic touches your chin. Meets both ANSI and Snell standards for impact protection. 7.5 oz. Lycra cover.	<BYTE value>	Minimum Chin Contact, Feather-Light, Maximum Protection Helmet
10032	110	SHM	Dense outer layer combines with softer inner layer to eliminate the mesh cover, no snagging on brush. Meets both ANSI and Snell standards for impact protection. 8.0 oz.	<BYTE value>	Mountain Bike Helmet: Smooth Cover Eliminates the Worry of Brush Snags But Delivers Maximum Protection
10033	110	HRO	Newest ultralight helmet uses plastic shell. Largest ventilation channels of any helmet on the market. 8.5 oz.	<BYTE value>	Lightweight Plastic with Vents Assures Cool Comfort Without Sacrificing Protection

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10034	110	HSK	Aerodynamic (teardrop) helmet covered with anti-drag fabric. Credited with shaving 2 seconds/mile from winner's time in Tour de France time-trial. 7.5 oz.	<BYTE value>	Teardrop Design Used by Yellow Jerseys, You Can Time the Difference
10035	111	SHM	Light-action shifting 10 speed. Designed for the city commuter with shock-absorbing front fork and drilled eyelets for carry-all racks or bicycle trailers. Internal wiring for generator lights. 33 lbs.	<BYTE value>	Fully Equipped Bicycle Designed for the Serious Commuter Who Mixes Business With Pleasure
10036	112	SHM	Created for the beginner enthusiast. Ideal for club rides and light touring. Sophisticated triple-buttet frame construction. Precise index shifting. 28 lbs.	<BYTE value>	We Selected the Ideal Combination of Touring Bike Equipment, then Turned It Into This Package Deal: High-Performance on the Roads, Maximum Pleasure Everywhere
10037	113	SHM	Ultra-lightweight. Racing frame geometry built for aerodynamic handlebars. Cantilever brakes. Index shifting. High-performance gearing. Quick-release hubs. Disk wheels. Bladed spokes.	<BYTE value>	Designed for the Serious Competitor, The Complete Racing Machine
10038	114	PRC	Padded leather palm and stretch mesh merged with terry back; Available in tan, black, and cream. Sizes S, M, L, XL.	<BYTE value>	Riding Gloves for Comfort and Protection

(6 of 12)

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10039	201	NKL	Designed for comfort and stability. Available in white & blue or white & brown. Specify size.	<BYTE value>	Full-Comfort, Long-Wearing Golf Shoes for Men and Women
10040	201	ANZ	Guaranteed waterproof. Full leather upper. Available in white, bone, brown, green, and blue. Specify size.	<BYTE value>	Waterproof Protection Ensures Maximum Comfort and Durability In All Climates
10041	201	KAR	Leather and leather mesh for maximum ventilation. Waterproof lining to keep feet dry. Available in white & gray or white & ivory. Specify size.	<BYTE value>	Karsten's Top Quality Shoe Combines Leather and Leather Mesh
10042	202	NKL	Complete starter set utilizes gold shafts. Balanced for power.	<BYTE value>	Starter Set of Woods, Ideal for High School and Collegiate Classes
10043	202	KAR	Full set of woods designed for precision control and power performance.	<BYTE value>	High-Quality Woods Appropriate for High School Competitions or Serious Amateurs
10044	203	NKL	Set of eight irons includes 3 through 9 irons and pitching wedge. Originally priced at \$489.00.	<BYTE value>	Set of Irons Available From Factory at Tremendous Savings: Discontinued Line
10045	204	KAR	Ideally balanced for optimum control. Nylon-covered shaft.	<BYTE value>	High-Quality Beginning Set of Irons Appropriate for High School Competitions
10046	205	NKL	Fluorescent yellow.	<BYTE value>	Long Drive Golf Balls: Fluorescent Yellow

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10047	205	ANZ	White only.	<BYTE value>	Long Drive Golf Balls: White
10048	205	HRO	Combination fluorescent yellow and standard white.	<BYTE value>	HiFlier Golf Balls: Case Includes Fluorescent Yellow and Standard White
10049	301	NKL	Super shock-absorbing gel pads disperse vertical energy into a horizontal plane for extraordinary cushioned comfort. Great motion control. Men's only. Specify size.	<BYTE value>	Maximum Protection For High-Mileage Runners
10050	301	HRO	Engineered for serious training with exceptional stability. Fabulous shock absorption. Great durability. Specify men's/women's, size.	<BYTE value>	Pronators and Supinators Take Heart: A Serious Training Shoe For Runners Who Need Motion Control
10051	301	SHM	For runners who log heavy miles and need a durable, supportive, stable platform. Mesh/synthetic upper gives excellent moisture dissipation. Stability system uses rear antipronation platform and forefoot control plate for extended protection during high-intensity training. Specify men's/women's size.	<BYTE value>	The Training Shoe Engineered for Marathoners and Ultra-Distance Runners
10052	301	PRC	Supportive, stable racing flat. Plenty of forefoot cushioning with added motion control. Women's only. D widths available. Specify size.	<BYTE value>	A Woman's Racing Flat That Combines Extra Forefoot Protection With a Slender Heel

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10053	301	KAR	Anatomical last holds your foot firmly in place. Feather-weight cushioning delivers the responsiveness of a racing flat. Specify men's/women's size.	<BYTE value>	Durable Training Flat That Can Carry You Through Marathon Miles
10054	301	ANZ	Cantilever sole provides shock absorption and energy rebound. Positive traction shoe with ample toe box. Ideal for runners who need a wide shoe. Available in men's and women's. Specify size.	<BYTE value>	Motion Control, Protection, and Extra Toebox Room
10055	302	KAR	Reusable ice pack with velcro strap. For general use. Velcro strap allows easy application to arms or legs.	<BYTE value>	Finally, an Ice Pack for Achilles Injuries and Shin Splints that You Can Take to the Office
10056	303	PRC	Neon nylon. Perfect for running or aerobics. Indicate color: Fluorescent pink, yellow, green, and orange.	<BYTE value>	Knock Their Socks Off With YOUR Socks
10057	303	KAR	100% nylon blend for optimal wicking and comfort. We've taken out the cotton to eliminate the risk of blisters and reduce the opportunity for infection. Specify men's or women's.	<BYTE value>	100% Nylon Blend Socks - No Cotton

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10058	304	ANZ	Provides time, date, dual display of lap/cumulative splits, 4-lap memory, 10 hr count-down timer, event timer, alarm, hour chime, waterproof to 50m, velcro band.	<BYTE value>	Athletic Watch w/4-Lap Memory
10059	304	HRO	Split timer, waterproof to 50m. Indicate color: Hot pink, mint green, space black.	<BYTE value>	Waterproof Triathlete Watch In Competition Colors
10060	305	HRO	Contains ace bandage, anti-bacterial cream, alcohol cleansing pads, adhesive bandages of assorted sizes, and instant-cold pack.	<BYTE value>	Comprehensive First-Aid Kit Essential for Team Practices, Team Traveling
10061	306	PRC	Converts a standard tandem bike into an adult/child bike. User-tested assembly instructions	<BYTE value>	Enjoy Bicycling With Your Child on a Tandem; Make Your Family Outing Safer
10062	306	SHM	Converts a standard tandem bike into an adult/child bike. Lightweight model.	<BYTE value>	Consider a Touring Vacation for the Entire Family: A Lightweight, Touring Tandem for Parent and Child
10063	307	PRC	Allows mom or dad to take the baby out too. Fits children up to 21 pounds. Navy blue with black trim.	<BYTE value>	Infant Jogger Keeps A Running Family Together
10064	308	PRC	Allows mom or dad to take both children! Rated for children up to 18 pounds.	<BYTE value>	As Your Family Grows, Infant Jogger Grows With You

(10 of 12)

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10065	309	HRO	Prevents swimmer's ear.	<BYTE value>	Swimmers Can Prevent Ear Infection All Season Long
10066	309	SHM	Extra-gentle formula. Can be used every day for prevention or treatment of swimmer's ear.	<BYTE value>	Swimmer's Ear Drops Specially Formulated for Children
10067	310	SHM	Blue heavy-duty foam board with Shimara or team logo.	<BYTE value>	Exceptionally Durable, Compact Kickboard for Team Practice
10068	310	ANZ	White. Standard size.	<BYTE value>	High-Quality Kickboard
10069	311	SHM	Swim gloves. Webbing between fingers promotes strengthening of arms. Cannot be used in competition.	<BYTE value>	Hot Training Tool - Webbed Swim Gloves Build Arm Strength and Endurance
10070	312	SHM	Hydrodynamic egg-shaped lens. Ground-in anti-fog elements; Available in blue or smoke.	<BYTE value>	Anti-Fog Swimmer's Goggles: Quantity Discount
10071	312	HRO	Durable competition-style goggles. Available in blue, grey, or white.	<BYTE value>	Swim Goggles: Traditional Rounded Lens For Greater Comfort

catalog_num	stock_num	manu_code	cat_descr	cat_picture	cat_advert
10072	313	SHM	Silicone swim cap. One size. Available in white, silver, or navy. Team Logo Imprinting Available.	<BYTE value>	Team Logo Silicone Swim Cap
10073	314	ANZ	Silicone swim cap. Squared-off top. One size. White	<BYTE value>	Durable Squared-off Silicone Swim Cap
10074	315	HRO	Re-usable ice pack. Store in the freezer for instant first-aid. Extra capacity to accommodate water and ice.	<BYTE value>	Water Compartment Combines With Ice to Provide Optimal Orthopedic Treatment

(12 of 12)

cust_calls Table

customer_num	call_dtime	user_id	call_code	call_descr	res_dtime	res_descr
106	1998-06-12 8:20	maryj	D	Order was received, but two of the cans of ANZ tennis balls within the case were empty.	1998-06-12 8:25	Authorized credit for two cans to customer, issued apology. Called ANZ buyer to report the QA problem.
110	1998-07-07 10:24	richc	L	Order placed one month ago (6/7) not received.	1998-07-07 10:30	Checked with shipping (Ed Smith). Order sent yesterday- we were waiting for goods from ANZ. Next time will call with delay if necessary.
119	1998-07-01 15:00	richc	B	Bill does not reflect credit from previous order.	1998-07-02 8:21	Spoke with Jane Akant in Finance. She found the error and is sending new bill to customer.
121	1998-07-10 14:05	maryj	O	Customer likes our merchandise. Requests that we stock more types of infant joggers. Will call back to place order.	1998-07-10 14:06	Sent note to marketing group of interest in infant joggers.

customer_num	call_dtime	user_id	call_code	call_descr	res_dtime	res_descr
127	1998-07-31 14:30	maryj	I	Received Hero watches (item # 304) instead of ANZ watches.		Sent memo to shipping to send ANZ item 304 to customer and pickup HRO watches. Should be done tomorrow, 8/1.
116	1997-11-28 13:34	mannyn	I	Received plain white swim caps (313 ANZ) instead of navy with team logo (313 SHM).	1997-11-28 16:47	Shipping found correct case in warehouse and express mailed it in time for swim meet.
116	1997-12-21 11:24	mannyn	I	Second complaint from this customer! Received two cases right-handed outfielder gloves (1 HRO) instead of one case lefties.	1997-12-27 08:19	Memo to shipping (Ava Brown) to send case of left-handed gloves, pick up wrong case; memo to billing requesting 5% discount to placate customer due to second offense and lateness of resolution because of holiday.

(2 of 2)

manufact Table

manu_code	manu_name	lead_time
ANZ	Anza	5
HSK	Husky	5
HRO	Hero	4
NRG	Norge	7
SMT	Smith	3
SHM	Shimara	30
KAR	Karsten	21
NKL	Nikolus	8
PRC	ProCycle	9

state Table

code	sname	code	sname
AK	Alaska	MT	Montana
AL	Alabama	NE	Nebraska
AR	Arkansas	NC	North Carolina
AZ	Arizona	ND	North Dakota
CA	California	NH	New Hampshire
CT	Connecticut	NJ	New Jersey
CO	Colorado	NM	New Mexico
DC	D.C.	NV	Nevada
DE	Delaware	NY	New York
FL	Florida	OH	Ohio

(1 of 2)

code	sname	code	sname
GA	Georgia	OK	Oklahoma
HI	Hawaii	OR	Oregon
IA	Iowa	PA	Pennsylvania
ID	Idaho	PR	Puerto Rico
IL	Illinois	RI	Rhode Island
IN	Indiana	SC	South Carolina
KY	Kentucky	TN	Tennessee
LA	Louisiana	TX	Texas
MA	Massachusetts	UT	Utah
MD	Maryland	VA	Virginia
ME	Maine	VT	Vermont
MI	Michigan	WA	Washington
MN	Minnesota	WI	Wisconsin
MO	Missouri	WV	West Virginia
MS	Mississippi	WY	Wyoming

(2 of 2)

The Sales_Demo Database

Your database server product contains scripts for the **sales_demo** database besides the **stores7** database described in previous sections. The **sales_demo** database provides an example of a simple data warehousing environment. The **sales_demo** database works in conjunction with the **stores7** database. The scripts for the **sales_demo** database create new tables and also add extra rows to the **items** and **orders** tables of the **stores7** database. In order to create the **sales_demo** database, you must first create the **stores7** database with the logging option. Once you create the **stores7** database, you can execute the scripts that create the **sales_demo** database from DB Access. The following figure gives an overview of the **sales_demo** database table.

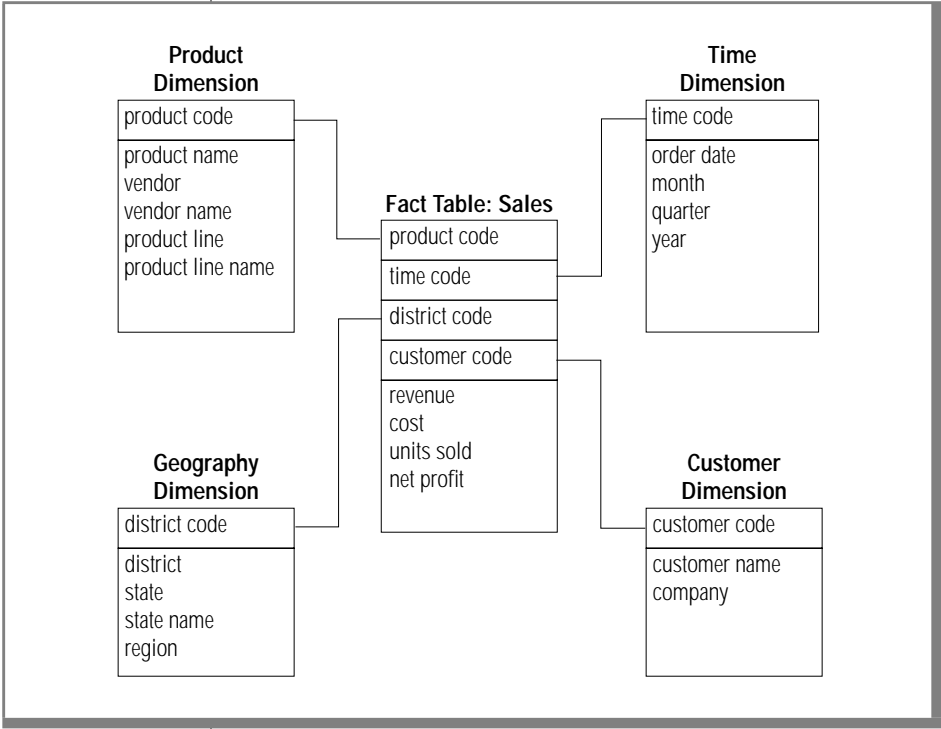


Figure 1-19
*The Completed
sales_demo Data
Model*

For detailed information on the structure of the **sales_demo** database and how to create it, see the [Informix Guide to Database Design and Implementation](#).

Glossary

access method	A procedure that is used to retrieve rows from or insert rows into a storage location. In the SET EXPLAIN statement, access method refers to the type of table access in a query (for example, SEQUENTIAL SCAN as opposed to INDEX PATH).
access privileges	The types of operations that a user has permission to perform in a specific database, table, table fragment, or column. Informix maintains its own set of database, table, table fragment, and column access privileges, which are independent of the operating-system access privileges.
active set	The collection of rows that satisfies a query associated with a cursor.
aggregate functions	The functions that return a single value based on the values of columns in one or more rows of a table (for example, the total number, sum, average, and maximum or minimum of an expression in a query or report). Aggregate functions are sometimes referred to as <i>set functions</i> or <i>math functions</i> .
alias	A temporary alternative name for a table in a query; usually used in complex subqueries and required for self-joins. In a form-specification file or any SQL query, alias refers to a single-word alternative name used in place of a more complex table name (for example, <i>t1</i> as an alias for <i>owner.table_name</i>).
alpha class	The alpha class of a code set consists of all characters that are classified as alphabetic. For example, the alpha class of the ASCII code set is the letters a-z and A-Z.
ANSI	Acronym for the American National Standards Institute. This group sets standards in many areas, including the computer industry and standards for SQL languages.

ANSI compliant	A database that conforms to certain ANSI standards. Informix databases can be created either as ANSI compliant or not ANSI compliant. An ANSI-compliant database enforces certain ANSI requirements, such as implicit transactions, required owner naming, and unbuffered logging (unbuffered logging only when using Informix Dynamic Server), that are not enforced in databases that are not ANSI compliant.
application development tool	Software, such as INFORMIX-NewEra, that you can use to create and maintain a database. The software allows a user to send instructions and data to and receive information from the database server.
application process	The process that manages an ESQL, NEWERA, or other program at runtime. It executes the program logic and initiates SQL requests. Memory that is allocated for program variables, program data, and the fetch buffer is part of this process. See also <i>database server process</i> .
application-productivity tools	Tools, such as forms and reports, used to write applications.
application program	An executable file or logically related set of files that perform one or more database management tasks.
application programming interface	See <i>SQL API</i> .
arbitrary rule	A series of expressions that you define using SQL relational and logical operators. Unlike the range rule, the arbitrary rule allows you to use any relational operator and any logical operator to define the expressions. Typically includes the use of the OR logical operator to group data.
archiving	Copying all the data and indexes of a database onto a new medium, usually a tape or a different physical device from the one that stores the database. Archived material is used for recovering from a failure and is usually performed by a database administrator. See <i>backup</i> .
argument	A value passed to a function, routine, stored procedure, or command.
array	A set of items of the same type. Individual members of the array are referred to as <i>elements</i> and are usually distinguished by an integer argument that gives the position of the element in the array. Informix arrays can have up to three dimensions.

ASCII	Acronym for the American Standards Committee for Information Interchange. Often used to describe an ordered set of printable and non-printable characters used in computers and telecommunication.
ASF	Acronym for Associated Services Facility. The code in the ASF portion of Informix products controls the connections between clients and servers. This term is used by system developers; users of Informix products see this term only in occasional error messages.
attached index	<p>Attached indexes are those that have no explicit fragmentation strategy. You create an attached index by omitting both the distribution scheme (specified by the FRAGMENT BY clause) and the storage option (specified by the IN clause) of the CREATE INDEX or ALTER FRAGMENT ON INDEX statements.</p> <p>The location of the index data varies depending on the database server. For Informix Dynamic Server database servers, index pages reside in the same tablespaces as the data pages they refer to.</p> <p>For Dynamic Server with UD Option, index pages for user indexes reside in separate tablespaces, but within the same dbspaces, as the data pages they refer to. Only the syscatalogs indexes reside in the same tablespace as the corresponding data pages.</p> <p>For Informix Dynamic Servers with AD and XP options, both user and system-catalog index pages reside in separate tablespaces but in the same dbspaces as the corresponding data pages.</p> <p>See <i>detached index</i></p>
audit event	(Not for Dynamic Server with AD and XP Options) Any database server activity or operation that could potentially access and alter data, which should be recorded and monitored by the database secure auditing facility. Examples of audit events include accessing tables, altering indexes, dropping chunks, granting database access, updating the current row, running database utilities, and so forth. (For a complete list of audit events, see Appendix A in the Trusted Facility Manual for your database server.)
audit file	(Not for Dynamic Server with AD and XP Options) A file that contains records of audit events and resides in the specified audit directory. Audit files form an audit trail of information that can be extracted by the database secure auditing facility for analysis by the database administrator.

audit mask	(Not for Dynamic Server with AD and XP Options) A structure that specifies which events should be audited by (or excluded from auditing by) the database secure auditing facility.
auxiliary statements	The SQL statements that you use to obtain auxiliary information about tables and databases. These statements include INFO, OUTPUT, WHENEVER, and GET DIAGNOSTICS.
B+ tree	A method of organizing an index into a tree structure for efficient record retrieval.
backup	A duplicate of a computer file on another device or tape to preserve existing work, in case of a computer failure or other mishap. In Informix Dynamic Server, <i>backup</i> refers to duplicating logical log files and <i>archiving</i> refers to duplicating data.
base table	See <i>table</i> .
before-image	The image of a row, page, or other item before any changes are made to it.
big-endian	A hardware-determined storage method in which the most-significant byte of a multibyte number has the lowest address. See little-endian.
blob	A legacy acronym for binary large object that is now known as and includes BYTE and TEXT data types. Blobs are data objects that effectively have no maximum size (theoretically as large as 2^{31} bytes).
blobpage	(Not for Dynamic Server with AD and XP Options) The unit of disk allocation within a blobspace under Informix Dynamic Server. The database server administrator determines the size of a blobpage. The size can vary, depending on the size of the BYTE or TEXT data that the user inserts.
blobspace	A logical collection of chunks that is used to store TEXT and BYTE data in a database server.
Boolean	A variable or an expression that can take on the logical values TRUE (1), FALSE (0), or UNKNOWN (if null values are involved).
buffer	A portion of computer memory where a program temporarily stores data. Data typically is read into or written out from buffers to disk.
buffered disk I/O	Disk I/O that is controlled by the operating system instead of an application. With buffered disk I/O, the operating system stores data in the kernel portion of memory before periodically writing the data to disk. See <i>unbuffered disk I/O</i> , <i>disk I/O</i> .

buffered logging	A type of logging that holds transactions in a memory buffer until the buffer is full, regardless of when the transaction is committed or rolled back. Informix database servers provide this option to speed up operations by reducing the number of disk writes.
byte	A unit of storage that corresponds approximately to one character. It is the smallest addressable computer memory unit. A byte is also known as an <i>octet</i> . (When BYTE appears in uppercase letters, it refers to the Informix data type.)
callback function	A user-defined function called during execution of an SQL request.
Cartesian product	The set that results when you pair each and every member of one set with each and every member of another set. A Cartesian product results from a multiple-table query when you do not specify the joining conditions among tables. See <i>join</i> .
cascading deletes	When any rows are deleted from the primary key column of a table, cascading deletes, if enabled, eliminate identical information from any foreign key column in a related table.
case-sensitivity	The condition of distinguishing between uppercase and lowercase letters. Be careful running Informix programs because certain commands and their options are case-sensitive; that is, they react differently to the same letters presented in uppercase and lowercase characters.
character	A logical unit of storage for the value code in a code set. It can be represented by one or more bytes and can be numeric, alphabetic, or a non-printable character (control character).
characterspecial device	See <i>unbuffered disk I/O</i> .
check constraint	A condition that must be met before data can be assigned to a table column during an INSERT or UPDATE statement.
child table	The referencing table in a referential constraint. See <i>parent table</i> .
chunk	(No blob space chunks for Dynamic Server with AD and XP Options) The largest contiguous section of disk space for a database server. A specified set of chunks defines a dbspace or blob space. A database server administrator allocates a chunk to a dbspace or blob space when that dbspace or blob space approaches full capacity.

client	An application program that requests services from a server program, typically a file server or a database server.
client locale	The environment that defines the behavior of the client application at runtime by specifying a language, code set, and the conventions used for a particular language, including date, time, and monetary formats.
client/server architecture	A hardware and software design that allows the user interface and database server to reside on separate nodes or platforms on a single computer or over a network.
client/server connection statements	The SQL statements that you use to make connections with databases. These statements include CONNECT, DISCONNECT, and SET CONNECTION.
close a cursor	To drop the association between a cursor and active set of rows that results from a query.
close a database	To deactivate the connection between a client and a database. Only one database can be active at a time.
close a file	To release the association between a file and a program.
cluster an index	To rearrange the physical data of a table according to a specific index.
cluster key	The column in a table that logically relates a group of blobs stored in an optical cluster.
clustersize	The amount of space, specified in kilobytes, that is allocated to an optical cluster on an optical volume.
code set	The representation of a national character set where each code value has a one-to-one mapping with a character graphic.
code-set conversion	The process of converting data from one code set to another when client and server computers use different code sets to represent the same character data.
cogroup	A named group of coservers. At initialization, the database server creates a cogroup that is named cogroup_all from all configured coservers.

collating sequence	The sequence of values that specifies some logical order in which the character fields in a database are sorted and indexed. Collation sequences may depend on order of the chosen code set (e.g. the order of characters and numbers in the ASCII code set) or it may depend on the locale chosen (e.g. the language, territory of the database, and further specifics as to whether a dictionary or a phone-book order is selected). Collating sequence is also known as <i>collation order</i> .
collocated join	A join that occurs locally on the coserver where the data resides. The local coserver sends the data to the other coservers after the join is complete.
column	A data element containing a particular type of information that occurs in every row of the table; also known as a <i>field</i> .
column expression	An expression that includes a column name and optionally uses column subscripts to define a column substring.
column subscript	A subscript in a column expression. See <i>subscript</i> .
column substring	A substring in a column expression. See <i>substring</i> .
command file	A system file that contains one or more statements or commands, such as SQL statements.
comment	The information in a program file that is not processed by the computer but documents the program. You use special characters such as a pound sign (#), curly braces ({ }), slash marks (/) and asterisks (*), or a double dash (--) to identify comments, depending on the programming environment.
COMMIT WORK	To complete a transaction by accepting all changes to the database since the beginning of the transaction.
COMMITTED READ	(An Informix level of isolation in which a user can view only rows that are currently committed at the moment of the query request; that is, a user cannot view rows that were changed as a part of a currently uncommitted transaction. COMMITTED READ is available through a database server and set with the SET ISOLATION statement. It is the default level of isolation for databases that are not ANSI compliant. See <i>isolation</i> .
compile	To translate a file that contains instructions (in a higher-level language) into a file containing the corresponding machine-level instructions.
compile-time errors	The errors that occur at the time the program source code is converted to executable form. Compare with <i>run-time errors</i> .

component	In the High-Performance Loader, the information required to load or unload data is organized in several <i>components</i> . The components are format, map, filter, query, project, device array, load job, and unload job.
composite index	An index constructed on two or more columns of a table. The ordering imposed by the composite index varies least frequently on the first-named column and most frequently on the last-named column.
composite join	A join between two tables based on the relationship among two or more columns in each table. See <i>join</i> .
compressed bitmap	An indexing method that identifies records through a fragment identifier and a record identifier.
concatenate	To append a second string to the end of a first string.
concatenation operator	A symbolic notation composed of two pipe symbols () used in expressions to indicate the joining of two strings.
concurrency	The ability of two or more processes to access the same database simultaneously.
configuration management (CM) coserver	A coserver that Informix designates to run CM software and store CM data.
configuration file	A file read during database server disk or shared-memory initialization that contains the parameters that specify values for configurable behavior. Database server and its archiving tool use configuration files.
connection	An association between an application and a database environment, created by a CONNECT or DATABASE statement. Database servers can also have connections to one another. See also <i>explicit connection</i> and <i>implicit connection</i> .
connection coserver	The coserver to which a client is directly connected. See <i>coserver</i> , <i>participating coserver</i> .
connection redirector	A Dynamic Server with AD and XP Options feature that is enabled by an option-field setting in the sqlhosts file whereby the database server attempts to establish a client connection with each coserver in a dbserver group until the connection is successful.
constant	A nonvarying data element or value.

constraint	A restriction on what kinds of data can be inserted or updated in tables. See also <i>check constraint</i> , <i>primary-key constraint</i> , <i>referential constraint</i> , <i>NOT NULL constraint</i> , and <i>unique constraint</i> .
control character	A character whose occurrence in a particular context initiates, modifies, or stops a control function (an operation to control a device, for example, in moving a cursor or in reading data). In a program, you can define actions that use the CTRL key with another key to execute some programming action (for example, entering CTRL-W to obtain on-line Help in Informix products). A control character is sometimes referred to as a <i>control key</i> . Compare to <i>printable character</i> .
cooked files	See <i>buffered disk I/O</i> .
correlation name	The prefix that you can use with a column name in a triggered action to refer to an old (before triggering statement) or a new (after triggering statement) column value. The associated column name must belong to the triggering table.
corrupted database	A database whose tables or indexes contain incomplete or invalid data.
corrupted index	An index that does not correspond exactly to its table.
coserver	The functional equivalent of a database server that operates on a single node. See <i>connection coserver</i> , <i>participating coserver</i> .
current row	The most recently retrieved row of the active set of a query.
cursor	An identifier associated with a group of rows; conceptually, the pointer to the current row. You can use cursors for SELECT statements or EXECUTE PROCEDURE statements (associating the cursor with the rows returned by a query) or INSERT statements (associating the cursor with a buffer to insert multiple rows as a group). A select cursor is declared for sequential only (regular cursor) or nonsequential (scroll cursor) retrieval of row information. In addition, you can declare a select cursor for update (initiating locking control for updated and deleted rows) or WITH HOLD (completing a transaction does not close the cursor). In ESQL/C, a cursor can be dynamic, meaning that it can be an identifier or a character/string variable.
cursor manipulation statements	The SQL statements that control cursors; specifically, the CLOSE, DECLARE, FETCH, FLUSH, OPEN, and PUT statements.

CURSOR STABILITY	An Informix level of isolation available through the SET ISOLATION statement in which the database server must secure a shared lock on a fetched row before the row can be viewed. The database server retains the lock until it receives a request to fetch a new row. See <i>isolation</i> .
data access statements	The SQL statements that you use to grant and revoke permissions and to lock tables.
data definition statements	The SQL statements that you use to create, alter, drop, and rename data objects, including databases, tables, views, synonyms, triggers, and stored procedures.
data dictionary	The collection of tables that keeps track of the structure of the database. Information about the database is maintained in the data dictionary, which is also referred to as the system catalog. See <i>system catalog</i> .
data distribution	A mapping of the data in a column into a set of the column values. The contents of the column are examined and divided into bins, each of which represents a percentage of the data. The organization of column values into bins is called the distribution for that column. You use the UPDATE STATISTICS statement to create data distributions.
data integrity	The process of ensuring that data corruption does not occur when multiple users simultaneously try to alter the same data. Locking and transaction processing are used to control data integrity.
data integrity statements	The SQL statements that you use to control transactions and audits. Data integrity statements also include statements for repairing and recovering tables.
data manipulation statements	The SQL statements that you use to query tables, insert into tables, delete from tables, update tables, and load into and unload from tables.
data restriction	Synonym for constraint. See <i>constraint</i> .
data type	A descriptor assigned to each column in a table or program variable, which indicates the type of data the column or program variable is intended to hold. Informix data types include SMALLINT, INTEGER, SERIAL, SMALLFLOAT, FLOAT, DECIMAL, MONEY, DATE, DATETIME, INTERVAL, CHAR, VARCHAR, TEXT, and BYTE. When GLS is enabled, Informix data types include NCHAR and NVARCHAR.

database	A collection of information (contained in tables) that is useful to a particular organization or used for a specific purpose.
Database Administrator	See <i>DBA</i> .
database application	A program that applies database management techniques to implement specific data manipulation and reporting tasks.
database environment	Used in the CONNECT statement, either the database server or the database server and database to which a user connects.
database locale	The environment that defines the language conventions and the behavior of read and write operations on the database.
database management system	See <i>DBMS</i> .
database object	In the broad sense, any SQL entity recorded in a system catalog table, such as a table, index, or stored procedure. In the restricted context of the SET statement, a database object is a constraint, index, or trigger whose name and current object mode are recorded in the sysobjstate system catalog table. You use the SET statement to change the object mode of database objects.
DBA	Acronym for <i>Database Administrator</i> . The DBA is the individual who is responsible for the contents and use of a database. This is different from the administrator of a database server, who is responsible for managing one or more database servers.
DBA-privileged	A class of stored procedures created only by a user with DBA database privileges.
DBMS	Acronym for <i>database management system</i> . It is all the components necessary to create and maintain a database, including the application development tools and the database server.
dbserver group	A collection of coservers defined and named by entries in the sqlhosts file. Dbserver groups make multiple coservers into a single logical entity for establishing or changing client/server connections.
dbslice	A named set of dbspaces that can span multiple coservers. A dbslice is managed as a single storage object. See <i>logslice</i> , <i>physslice</i> , <i>rootslice</i> .
DDL	Acronym for data definition language. See <i>data definition statements</i> .

deadlock	A situation in which two or more threads cannot proceed because each is waiting for data locked by the other (or another) thread. Informix Dynamic Server monitors and prevents potential deadlock situations by sending an error message to the application whose request for a lock may result in a deadlock.
debug file	A file that receives output used for debugging purposes.
decision-support application	An application that provides information which is used for strategic planning, decision-making, and reporting. It typically executes in a batch environment in a sequential scan fashion and returns a large fraction of the rows scanned. Decision-support queries typically scan the entire database. See <i>online transaction processing application</i> .
decision-support query	A query that is generated by a decision-support application. It often requires operations such as multiple joins, temporary tables, and extensive calculations, and can benefit significantly from PDQ. Also see <i>OLTP query</i> .
declaration statement	A programming language statement that describes or defines objects, for example, defining a program variable. Compare to <i>procedural statement</i> .
default	How a program acts or the values that are assumed if the user does not explicitly specify an action.
default value	A value inserted into a column or variable when an explicit value is not specified. Default values can be assigned to columns using the ALTER TABLE and CREATE TABLE statements and to variables in stored procedures.
delete	To remove any row or combination of rows with the DELETE statement.
delimited identifier	An identifier surrounded by double quotes. The purpose of a delimited identifier is to allow usage of identifiers that are otherwise identical to SQL reserved keywords or that contain non-alphabetical characters. See also <i>identifier</i> .
delimiter	A boundary on an input field or the terminator for a column or row. Some files and prepared objects require semicolon (;), comma (,), space, or tab delimiters between statements.
deluxe mode	Dynamic Server with AD and XP Options method of loading or unloading data that uses regular inserts.

descriptor	A quoted string or embedded variable name that identifies an allocated system-descriptor area or an sqllda structure. It is used for the Informix SQL APIs. See <i>identifier</i> .
detached index	<p>The type of index you get when the distribution scheme (specified by the FRAGMENT BY clause), and the storage option (specified by the IN clause) of the CREATE INDEX or ALTER FRAGMENT ON INDEX statements differs from the distribution scheme of the underlying table.</p> <p>Index pages reside in separate dbspaces from the corresponding data pages.</p>
device array	A list of I/O devices. See <i>component</i> .
diagnostic area	A data structure that stores diagnostic information about an executed SQL statement.
diagnostics table	A special table that holds information about the integrity violations caused by each row in a violations table. You use the START VIOLATIONS TABLE statement to create violations and diagnostics tables and associate them with a base table.
disabled mode	The object mode in which a database object is disabled. When a constraint, index, or trigger is in the disabled mode, the database server acts as if the object does not exist and does not take it into consideration during the execution of data manipulation statements.
disk configuration	The organization of data on a disk; also refers to the process of preparing a disk to store data.
disk I/O	The process of transferring data between memory and disk. The I/O refers to input/output.
display label	A temporary name for a column or expression in a query.
distribution	See <i>data distribution</i> .
DML	Acronym for data manipulation language. See <i>data manipulation statements</i> .
dominant table	See <i>outer join</i> .
DRDA	Acronym for Distributed Relational Database Architecture. DRDA is an IBM-defined set of protocols that software manufacturers can follow to develop connectivity solutions between heterogeneous relational database management environments.

DSS	Decision Support System. See <i>decision support application</i>
duplicate index	An index that allows duplicate values in the indexed column.
dynamic management statements	The SQL statements that describe, execute, and prepare statements.
dynamic SQL	The statements and structures that allow a program to form an SQL statement during execution, so that portions of the statement can be determined by user input.
dynamic statements	The SQL statements that are created at the time the program is executed, rather than when the program is written. You use the PREPARE statement to create dynamic statements.
EBCDIC	Extended Binary Coded Decimal Interchange Code. An 8-bit, 256-element character set.
embedded SQL	The SQL statements that are placed within a host language. Informix supports embedded SQL in C.
enabled mode	The default object mode of database objects. When a constraint, index, or trigger is in this mode, the database server recognizes the existence of the object and takes the object into consideration while executing data manipulation statements. Also see <i>object mode</i> .
environment variable	A variable that is maintained by the operating system for each user and made available to all programs that the user runs.
error log	A file that receives error information whenever a program runs.
error message	A message that is associated with a (usually negative) designated number. Informix applications display error messages on the screen or write them to files.
error trapping	The code within a program that anticipates and reacts to run-time errors.
escape character	A character that indicates that the following character, normally interpreted by the program, is to be printed as a literal character instead. The escape character is used with the interpreted character to “escape” or ignore the interpreted meaning.

escape key	The keyboard key, usually marked ESC, that is used to terminate one mode and start another mode in most UNIX and DOS systems.
exception	An error or warning returned by the database server or a state initiated by a stored procedure statement.
exclusive access	When a user has sole access to a database or table and other users are prevented from using it.
exclusive lock	A lock on an object (row, page, table, or database) that is held by a single thread that prevents other processes from acquiring a lock of any kind on the same object.
executable file	A binary file containing compiled code that can be run as a program; can also refer to a UNIX shell script or an MS-DOS batch file.
execute	To carry out a program, procedure, or a set of instructions.
explicit connection	A connection made to a database environment that uses the CONNECT statement.
exponent	The power to which a value is to be raised.
express mode	A Dynamic Server with AD and XP Options method of loading or unloading data that uses light appends.
expression	Anything from a simple numeric or alphabetic constant to a more complex series of column values, functions, quoted strings, operators, and keywords. A Boolean expression contains a logical operator (>, <, =, !=, IS NULL, and so on) and evaluates as TRUE, FALSE, or UNKNOWN. An arithmetic expression contains the operators (+, -, ×, /, and so on) and evaluates as a number.
expression-based distribution scheme	User-defined distribution scheme created by formulating a series of fragment expressions to be used by a database server to distribute rows to fragments.
extent	(A continuous segment of disk space that a database server allocated to a tblspace (a table). The user can specify both the initial extent size for a table and the size of all subsequent extents that a database server allocates to the table.
external table	A database table that is not in the current database. It may or may not be in a database managed by the same database server.

family name	A quoted string constant that specifies a family name in the optical family. See <i>optical family</i> .
fault tolerance	See <i>high availability</i> .
fetch	The action of moving a cursor to a new row and retrieving the row values into memory.
fetch buffer	A buffer in the application process that the database server uses to send fetched row data (except BYTE and TEXT data) to the application. See also <i>application process</i> .
field	See <i>column</i> .
file	A collection of related information stored together on a system, such as the words in a letter or report, a computer program, or a listing of data.
file server	A network node that manages a set of disks and provides storage services to computers on the network.
filename extension	The part of a filename following the period. For example, DB-Access appends the extension .sql to command files.
filter	A set of conditions for selecting records from an input file. See <i>component</i> .
filtering mode	The object mode of a database object that causes bad rows to be filtered out to the violations table during data manipulation operations. Only constraints and unique indexes can be in the filtering mode. When a constraint or unique index is in this mode, the database server enforces the constraint or the unique index requirement during INSERT, DELETE, and UPDATE operations but filters out rows that would violate the constraint or unique index requirement.
fixchar	A character data type, available in ESQL/C programs, in which the character string is fixed in length, padded with trailing blanks if necessary, and not null-terminated.
fixed-point number	A number where the decimal point is fixed at a specific place regardless of the value of the number.
flag	A command-line option, usually indicated by a minus (-) sign in UNIX systems. For example, in DB-Access the -e flag echoes input to the screen.
flexible temporary table	An explicit temporary table that Dynamic Server with AD and XP Options automatically fragments using a round-robin distribution scheme.

floating-point number	A number with fixed precision (total number of digits) and undefined scale (number of digits to the left of the decimal point). The decimal point <i>floats</i> as appropriate to represent an assigned value.
foreign key	A column, or set of columns, that references a unique or primary key in a table. For every entry in a foreign-key column, there must exist a matching entry in the unique or primary column, if all foreign-key columns contain non-null values.
format	A description of the organization of a data file. See <i>component</i> .
formatting character	A percent-sign (%) followed by a letter (c, n, o, or r). When used in a command line, Informix Dynamic Server with AD and XP options expands the formatting character to designate multiple coserver numbers (%c), multiple nodes (%n), multiple ordinal numbers designating dbspaces (%d), or a range of dbspaces (%r).
fragment	See <i>index fragment</i> and <i>table fragment</i> .
fragmentation	The process of defining groups of rows within a table based on a rule and then storing these groups, or fragments, in separate dbspaces that you specify when you create a table or index fragmentation strategy.
function	See <i>program block</i> .
gateway	A data communications device that establishes communications between networks.
gigabyte	Gigabyte is a unit of storage. A gigabyte equals 1024 megabytes or 1024^3 bytes.
Global Language Support (GLS)	An application environment that allows Informix APIs and database servers to handle different languages, cultural conventions, and code sets.
global variable	A variable whose value you can access from any module or function in a program or stored procedure. See <i>variable</i> and <i>scope of reference</i> .
GLS	See <i>Global Language Support</i> .
help message	On-line text displayed automatically or at the request of the user to assist the user in interactive programs. Such messages are stored in help files.
hierarchy	A tree-like data structure in which some groups of data are subordinate to others such that only one group (called <i>root</i>) exists at the highest level, and each group except root is related to only one parent group on a higher level.

high availability	The ability of a system to resist failure and data loss. High availability includes features such as fast recovery and mirroring. It is sometimes referred to as <i>fault tolerance</i> .
High-Performance Loader	The High-Performance Loader utility is part of Informix Dynamic Server. The HPL loads and unloads data using parallel access to devices. See <i>external table</i> .
highlight	A rectangular inverse-video area that marks your place on the screen. A highlight often indicates the current option on a menu or the current character in an editing session. If a terminal cannot display highlighting, the current option often appears in angle brackets, and the current character is underlined.
hold cursor	A cursor that is created using the WITH HOLD keywords. A hold cursor remains open past the end of a transaction. It allows uninterrupted access to a set of rows across multiple transactions.
home page	The page that contains the first byte of the data row. Even if a data row outgrows its original storage location, the home page does not change. The home page contains a forward pointer to the new location of the data row. See <i>remainder page</i> .
host variable	A C program variable that is referenced in an embedded statement. A host variable is identified by the dollar sign (\$) or colon (:) that precedes it.
HPL	See <i>High-Performance Loader</i> .
identifier	A sequence of letters, digits, and underscores (_) that represents the name of a database, table, column, cursor, function, index, synonym, alias, view, prepared object, constraint, or procedure name.
implicit connection	A connection made using a database statement (DATABASE, CREATE DATABASE, START DATABASE, DROP DATABASE). Also see <i>explicit connection</i> .
incremental archiving	A system of archiving that allows the option to archive only those parts of the data that have changed since the last archive was created.
index	A structure of pointers that point to rows of data in a table. An index optimizes the performance of database queries by ordering rows to make access faster.

index fragment	Consists of zero or more index items grouped together, which can be stored in the same dbspace as the associated table fragment or, if you choose, in a separate dbspace.
Informix user ID	A login user ID (login user name) that must be valid on all computer systems (operating systems) involved in the client's database access. Often referred to as the client's user ID or user name. The user ID does not need to refer to a fully functional user account on the computer system; only the user identity components of the user account information is significant to Informix database servers. Any given user typically has the same Informix user ID on all networked computer systems involved in the database access.
Informix user password	A user ID password that must be valid on all computer systems (operating systems) involved in the client's database access. When the client specifies an explicit user ID, most computer systems require the Informix user password to validate the user ID.
initialize	To assign a starting value to a variable or to start operations.
inmigration	The process by which Optical Subsystem migrates BYTE or TEXT data from the optical storage subsystem into the Informix Dynamic Server environment.
input	The information that is received from an external source (for example, from the keyboard, a file, or another program) and processed by a program.
input parameter	A placeholder within a prepared SQL statement that indicates a value is to be provided at the time the statement is executed.
insert cursor	A cursor for insert operations. Allows bulk insert data to be buffered in memory and written to disk.
installation	The loading of software from some magnetic medium (tape, cartridge, or floppy disk) or CD onto a computer and preparing it for use.
interactive	Refers to a program that accepts input from the user, processes the input, and then produces output on the screen, in a file, or on a printer.
interrupt	A signal from a user or another process that can stop the current process temporarily or permanently. See <i>signal</i> .

interrupt key	A key used to cancel or abort a program or to leave a current menu and return to the menu one level above. On many systems, the CONTROL-C keypress is used for the interrupt key. On other systems, the interrupt key is DEL or CONTROL-Break.
intraquery parallelization	Breaking of a single query by a database server using PDQ into subqueries and then processing the subqueries in parallel. Parallel processing of this type has important implications when each subquery retrieves data from a fragment of a table. Because each partial query operates on a smaller amount of data, the retrieval time is significantly reduced. Also see <i>interquery parallelization</i> .
IPX/SPX	Acronym for Internetwork Packet Exchange/Sequenced Packet Exchange. It refers to the NetWare network protocol by Novell.
ISAM	Acronym for Indexed Sequential Access Method. An indexed sequential access method allows you to find information in a specific order or to find specific pieces of information quickly through an index. See <i>access method</i> .
ISAM error	Operating system or file access error.
ISO	Acronym for the International Standards Organization. ISO sets worldwide standards for the computer industry, including standards for character input and manipulation, code sets, and SQL syntax.
join	The process of combining information from two or more tables based on some common domain of information. Rows from one table are paired with rows from another table when information in the corresponding rows match on the joining criterion. For example, if a customer_num column exists in the customer and the orders tables, you can construct a query that pairs each customer row with all the associated orders rows based on the common customer_num . See <i>Cartesian product</i> and <i>outer join</i> .
jukebox	A cabinet that consists of one or more optical-disc drives, slots that store optical platters when they are not mounted, and a robotic arm that transfers platters between the slots and the drives. A jukebox is also known as an <i>autochanger</i> .
kernel	The part of the operating system that controls processes and the allocation of resources.

key	The pieces of information that are used to locate a row of data. A key defines the pieces of information you want to search for, as well as the order in which you want to process information in a table. For example, you can index the last_name column in a customer table to find specific customers or to process the customers in alphabetical or reverse alphabetical order, according to their last names (last_name serves as the key).
keyword	A word that has meaning to a program. For example, the word SELECT is a keyword in SQL.
kilobyte	Kilobyte is a unit of storage. A kilobyte equals 1024 bytes.
language supplement	The result of the product localization process. A language supplement for a specific Western European language can be installed with an Informix product to allow the user to see error and warning messages in a language other than U.S. English. If installed with DB-Access, the menu names and options and on-line Help for that product also appear in the specified language.
level of isolation	See <i>isolation</i> .
library	A collection of precompiled functions or routines that are designed to perform tasks which are common to a particular kind of application. Your product can include library functions or routines that you can call from your programs.
light append	An unbuffered, unlogged insert operation.
link	The process of combining separately compiled program modules into an executable program.
literal	A character or numeric constant.
little-endian	A hardware-determined storage method in which the least- significant byte of a multibyte number has the lowest address. See big-endian.
load job	The information required to load data into a Dynamic Server database using the HPL. This information includes format, map, filter, device array, project, and special options.
local character	A character in a native language character set; also known as a <i>national character</i> or a <i>native character</i> .
local loopback	A connection between the client and database server that uses a network connection even though the client and the database servers are on the same computer.

local variable	A variable that has meaning only in the module in which it is defined. See <i>variable</i> and <i>scope of reference</i> .
locale	The environment that defines the behavior of the program at runtime by specifying a language, code set, and local customs. It usually is based on the linguistic customs and rules of the region or territory. The locale can be expressed through an environment variable setting that dictates output formats for numbers, currency symbols, dates, and time or collation order for character strings and regular expressions.
locking	The process of temporarily limiting access to an object (database, table, page, or row) to prevent conflicting interactions among concurrent processes. Locks can be in either exclusive mode, which restricts read and write access to only one user; or share mode, which allows read-only access to other users. In addition, update locks exist that begin in share mode but are upgraded to exclusive mode when a row is changed.
locking granularity	The size of a locked object. The size may be a database, table, page, or row.
logical log	An allocation of disk space managed by the database server that contains records of all changes that were performed on a database during the period the log was active. The logical log is used to roll back transactions, recover from system failures, and restore databases from archives.
login	The process of identifying oneself to a computer.
login password	See <i>Informix user password</i> .
login user ID	See <i>Informix user ID</i> .
logslice	A dbslice whose contents are comprised solely of logical-log files. The logical-log files in the logslice can be owned by multiple coservers, one log file per dbspace. See <i>dbslice</i> , <i>rootslice</i> , <i>physslice</i> .
macro	A named set of instructions that the computer executes whenever the name is referenced.
mantissa	The significant digits in a floating-point number.
map	A description of the relation between the records of a data file and the columns of a Dynamic Server database. See <i>component</i> .

massively parallel processing system	A massively parallel processing (MPP) system is composed of multiple computers that are connected to a single high-speed communication subsystem. The computers can be partitioned into nodes.
math functions	See <i>aggregate functions</i> .
Memory Grant Manager (MGM)	(Not for Dynamic Server with AD and XP Options) A database server component that coordinates the use of memory and I/O bandwidth for decision-support queries. MGM uses the DS_MAX_QUERIES, DS_TOTAL_MEMORY, DS_MAX_SCANS, and PDQPRIORITY configuration parameters to determine what resources can or cannot be granted to a decision-support query.
megabyte	Megabyte is a unit of storage. A megabyte equals 1024 kilobytes or 1024 ² bytes.
menu	A screen display that allows you to choose the commands that you want the computer to perform.
mirroring	Storing the same data on two chunks simultaneously. If one chunk fails, the data is still usable on the other chunk in the mirrored pair. This data storage option is handled by the database server administrator.
MODE ANSI	The keywords specified on the CREATE DATABASE statement to make a database ANSI compliant.
monochrome	A term that describes a monitor that can display only one color.
MPP system	See <i>massively parallel processing system</i> .
multibyte character	A character that requires more than one byte to represent it.
multithreading	Running of multiple threads that are run within the same process. Also see <i>thread</i> .
national character	See <i>local character</i> .

node	<p>Within the context of an index for a database, a node is an ordered group of key values having a fixed number of elements. (A key is a value from a data record.) A B+ tree, for example, is a set of nodes that contain keys and pointers that are arranged in a hierarchy.</p> <p>Within the context of an MPP system, a node is an individual computer. See <i>massively parallel processing system</i>.</p> <p>Within the context of an SMP system, a node can either be the entire SMP computer or a fully functioning subsystem that uses a portion of the hardware resources of that SMP system. See <i>symmetric-multiprocessing system</i>.</p>
not null constraint	A constraint on a column that specifies the column cannot contain null values.
null value	A value that is unknown or not applicable. (A null is not the same as a value of zero or blank.)
object	See <i>database object</i> .
object mode	The state of a database object as recorded in the sysobjstate system catalog table. A constraint or unique index can be in the enabled, disabled, or filtering mode. A trigger or duplicate index can be in the enabled or disabled mode. You use the SET statement to change the object mode of an object.
OLTP	Online Transaction Processing. See <i>online transaction processing application</i> .
online transaction processing application	Characterized by quick, indexed access to a small number of data items. The applications are typically multiuser, and response times are measured in fractions of seconds. See <i>decision support application</i>
online transaction processing queries	The transactions handled by OLTP applications are usually simple and pre-defined. A typical OLTP system is an order-entry system where only a limited number of rows is accessed by a single transaction many times.
Dynamic Server instance	The set of processes, storage spaces, and shared memory that together comprise a complete database server.
ON-Monitor	(Not for Dynamic Server with AD and XP Options) An interface that presents a series of screens through which a database server administrator can monitor and modify a database server.

open	The process of making a resource available, such as preparing a file for access, activating a cursor, or initiating a window.
operational table	A logging permanent table that uses light appends for fast update operations. Operational tables do not perform record-by-record logging.
optical cluster	An amount of space, on an optical disc, that is reserved for storing a group of logically related blobs.
optical family	A group of optical discs, theoretically unlimited in number.
optical platters	The removable optical discs that store data in an optical storage subsystem.
optical statements	The SQL statements that you use to control optical clustering.
optical volume	One side of a removable Write-Once-Read-Many (WORM) optical disc.
outer join	An asymmetric joining of a dominant and a subordinate table in a query; the joining restrictions apply only to the subordinate or <i>outer</i> table. Rows in the dominant table are retrieved without considering the join, but rows from the outer table are included only if they also satisfy the join conditions. Any dominant-table rows that do not have a matching row from the outer table receive a row of nulls in place of an outer-table row.
outmigration	The process by which Optical Subsystem migrates BYTE or TEXT data from the Informix Dynamic Server environment to an optical storage subsystem.
output	The result that the computer produces in response to a query or a request for a report.
owner-privileged	A class of stored procedures that can be created by any user who has Resource database privileges.
packed decimal	A storage format that represents either two decimal digits or a sign and one decimal digit in each byte.
pad	Usually a space or blank to fill empty places at the beginning or end of a line, string, or field.
parallel database query	The execution of SQL queries in parallel rather than sequential order. The tasks a query requires are distributed across several processors. This type of distribution enhances database performance.

parallel-processing platform	A parallel-processing platform is a set of independent computers that operate in parallel and communicate over a high-speed network, bus, or interconnect. See <i>symmetric multiprocessing computer</i> , <i>massively parallel processing system</i> .
parallelism	Ability of an Informix database server to process a task in parallel by breaking the task into subtasks and then processing them simultaneously.
parameter	A variable that is given a constant value for a specified application. In a subroutine, a parameter is the placeholder for the argument values that are passed to the subroutine at runtime.
parent-child relationship	See <i>referential constraint</i> .
parent table	The referenced table in a referential constraint. See <i>child table</i> .
partial character	A multibyte character that has lost one or more bytes so that the original intended meaning of the character is lost. GLS (Global Language Support) software provides context-specific solutions that prevent partial characters from being generated during string-processing operations.
participating coserver	A coserver that controls one or more fragments of a table that Dynamic Server with AD and XP Options accesses. See <i>coserver</i> , <i>connection coserver</i> .
partition	See <i>table fragment</i> .
pattern	An identifiable or repeatable series of characters or symbols.
PDQ	See <i>parallel database query</i> .
PDQ priority	Determines the amount of resources that a database server allocates to process a query in parallel. These resources include memory, threads (such as scan threads), and sort space. The level of parallelism is established by using the PDQPRIORITY environment variable or various database server configuration parameters (including PDQPRIORITY and MAX_PDQPRIORITY) or dynamically through the SET PDQPRIORITY statement.
permission	On some operating systems, the right to access files and directories.
phantom row	A row of a table that is initially modified or inserted during a transaction but is subsequently rolled back. Another user can see a phantom row if the isolation level is Informix DIRTY READ or ANSI READ UNCOMMITTED. No other isolation levels allow a changed but uncommitted row to be seen.

physslice	A dbslice that contains the physical log. See <i>dbslice</i> , <i>logslice</i> , <i>rootslice</i> .
pointer	A number that specifies the “address-in-memory” of the data or variable of interest.
precision	The total number of significant digits in a real number, both to the right and left of the decimal point. For example, the number 1437.2305 has a precision of 8. See <i>scale</i> .
prepared statement	An SQL statement that is generated by the PREPARE statement from a character string or from a variable containing a character string. This feature allows you to form your request while the program is executing without having to modify and recompile the program.
preprocessor	A program that takes high-level programs and produces code that a standard language compiler, such as C can compile.
primary key	The information from a column or set of columns that uniquely identifies each row in a table. The primary key sometimes is called a <i>unique key</i> .
primary-key constraint	Specifies that each entry in a column or set of columns contains a non-null unique value.
printable character	A character that can be displayed on a terminal, screen, or printer. Printable characters include A-Z, a-z, 0-9, and punctuation marks. Compare with <i>control character</i> .
privilege	The right to use or change the contents of a database, table, table fragment, or column. See <i>access privileges</i> .
procedural statement	A programming language statement that specifies actions; for example, looping and branching if a condition is met. Compare with <i>declaration statement</i> .
procedure	See <i>program block</i> and <i>stored procedure</i> .
procedure cursor	A cursor that is associated with an EXECUTE PROCEDURE statement. It enables you to scan multiple rows of data, moving data row by row into a set of receiving variables.
process	A discrete task, generally a program, that the operating system executes.
program block	A named collection of statements, such as a function, routine, or procedure, that performs a particular task; a unit of program code.
program control	The actions that a computer takes, as opposed to actions that a user takes.

project	A group of related components used by the HPL. See <i>component</i> .
projection	Taking a subset from the columns of a single table. Projection is implemented through the select list in the SELECT clause of a SELECT statement and returns some of the columns and all the rows in a table. See <i>selection</i> and <i>join</i> .
promotable lock	A lock that can be changed from a shared lock to an exclusive lock. See <i>update lock</i> .
protocol	A set of rules that govern communication among computers. These rules govern format, timing, sequencing, and error control.
query	A request to the database to retrieve data that meets certain criteria, usually with the SELECT statement. When used with the High Performance Loader, selects records to unload from a Dynamic Server database. See <i>component</i> .
query optimization information statements	The SQL statements that are used to optimize queries. These statements include SET EXPLAIN, SET OPTIMIZATION, and UPDATE STATISTICS.
query unnesting	An execution strategy for nested SQL subqueries whereby Dynamic Server with AD and XP Options rewrites such subqueries to use modified joins rather than iteration mechanisms. The sqexplain.out file reflects the query plan that has been selected after subquery unnesting has occurred.
range rule	A user-defined algorithm that you use to define the boundaries of each fragment in a table using SQL relational and logical operators. Expressions in a range rule can use the following restricted set of operators: >, <, >=, <=, and the logical operator AND.
raw table	A nonlogged permanent table that uses light appends.
raw device	See <i>unbuffered disk I/O</i> .
raw disk	See <i>unbuffered disk I/O</i> .
Read Committed	An ANSI level of isolation available through Informix Dynamic Server and set with the SET TRANSACTION statement in which a user can view only rows that are currently committed at the moment of the query request; that is, a user cannot view rows that were changed as a part of a currently uncommitted transaction. It is the default level of isolation under Dynamic Server for databases that are not ANSI COMPLIANT. See <i>isolation</i> .

Read Uncommitted	An ANSI level of isolation set with the SET TRANSACTION statement that does not account for locks and allows viewing of any existing rows, even rows that currently can be altered from inside an uncommitted transaction. Read Uncommitted is the lowest level of isolation (no isolation at all). See <i>isolation</i> .
real user ID	See <i>Informix user ID</i> .
record	See <i>row</i> .
Record-ID	A four-byte RSAM entity describing the logical position of the record within a fragment. Also referred to as RID, but not to be confused with Row-ID
recover a database	To restore a database to a former condition after a system failure or other destructive event. The recovery restores the database as it existed immediately before the crash. This is sometimes referred to as <i>restore a database</i> .
referential constraint	The relationship between columns within a table or between tables; also known as a <i>parent-child relationship</i> . Referencing columns also are known as <i>foreign keys</i> .
relation	See <i>table</i> .
relational database	A database that uses table structures to store data. Data in a relational database is divided across tables in such a way that additions and modifications to the data can be made easily without loss of information.
remote	A connection that requires a network.
Repeatable Read	An Informix and ANSI level of isolation available through Dynamic Server with the Informix SET ISOLATION statement or ANSI SET TRANSACTION statement, which ensures all data read during a transaction is not modified during the entire transaction. Transactions under ANSI Repeatable Read are also known as Serializable. Informix Repeatable Read is the default level of isolation under Dynamic Server for ANSI-compliant databases. See <i>isolation</i> and <i>Serializable</i> .
reserved word	A word in a statement or command that you cannot use in any other context of the language or program without receiving a warning or error message.
restore a database	See <i>recover a database</i> .
role	A classification or work task, such as payroll , assigned by the DBA. Assignment of roles makes management of privileges convenient.

role separation	(Not for Dynamic Server with AD and XP Options) A database server installation option that allows different users to perform different administrative tasks.
roll back	The process that reverses an action or series of actions on a database. The database is returned to the condition that existed before the actions were executed. See <i>transaction</i> .
rootslice	A dbslice that contains the root dbspaces for all coservers. See <i>dbslice</i> , <i>logslice</i> , <i>physlice</i> .
round-robin distribution scheme	A data distribution scheme. A database server distributes rows in such a way that the number of rows in each of the fragments remains approximately the same.
routine	See <i>program block</i> .
row	A group of related items of information about a single entity in a database table. In a table of customer information, for example, a row contains information about a single customer. A row is sometimes referred to as a <i>record</i> or <i>tuple</i> . (In a screen form, a row can refer to a line of the screen.)
rowid	In nonfragmented tables, rowid refers to an integer that defines the physical location of a row. Rowids must be explicitly created to be used in fragmented tables and they do not define a physical location for a row. Rowids in fragmented tables are accessed by an index that is created when the rowid is created; this access method is slow. Informix recommends that users creating new applications move toward using primary keys as a method of row identification instead of using rowids.
rule	How a database server or a user determines into which fragment rows are placed. The database server determines the rule for a round-robin-based distribution scheme. The user determines the rule for an expression-based distribution scheme.
run-time environment	The hardware and operating-system services available at the time a program runs.
run-time errors	Errors that occur during program execution. Compare with <i>compile-time errors</i> .
scale	The number of digits to the right of the decimal place in DECIMAL notation. The number 14.2350 has a scale of 4 (four digits to the right of the decimal point). See <i>precision</i> .

scale up	The ability to compensate for an increase in query size by adding a corresponding amount of computer resources so that processing time does not also increase.
scan thread	A database server thread that is assigned the task of reading rows from a table. When a query is executed in parallel, the database server allocates multiple scan threads to perform the query in parallel.
schema	The structure of a database or a table. The schema for a table lists the names of the columns, their data types, and (where applicable) the lengths, indexing, and other information about the structure of the table.
scope of reference	The portion of a program in which an identifier applies and can be accessed. Three possible scope sizes exist: local (an identifier applies only within a single program block), modular (the identifier applies throughout a single module), and global (an identifier applies throughout the entire program).
scratch table	A nonlogging temporary table.
scroll cursor	A cursor created with the SCROLL keyword that allows you to fetch rows of the active set in any sequence.
secure auditing	(Not for Dynamic Server with AD and XP Options) A facility of Informix database servers that lets a database server administrator keep track of unusual or potentially harmful user activity. Use the onaudit utility to enable auditing of events and create audit masks, and the onshowaudit utility to extract the audit event information for analysis.
select	See <i>query</i> .
select cursor	A cursor associated with the SELECT statement. It enables you to scan multiple rows of data, moving data row by row into a set of receiving variables.
selection	Taking a horizontal subset of the rows of a single table that satisfies a particular condition. Selection is implemented through the WHERE clause of a SELECT statement and returns some of the rows and all of the columns in a table. See <i>projection</i> and <i>join</i> .
self-join	A join between a table and itself. A self-join occurs when a table is used two or more times in a SELECT statement (with different aliases) and joined to itself.

semaphore	An operating-system communication device that signals a process to awaken.
server locale	The environment that defines the locale that the database server uses when it performs read and write operations on the database server computer.
server name	The unique name of a database server. The database server name is used by an application to select a database server. It is assigned by the administrator of the database server.
server processing locale	The environment that the database server dynamically determines based on the client locales and information that is stored in the database being accessed.
set functions	See <i>aggregate functions</i> .
shared library	Like a standard or static library, a shared library contains routines that are used by applications in the course of processing data. Unlike static libraries, shared libraries are not linked at compile time to the application, but instead are loaded into memory by the operating system as they are needed. The copy of the library that the operating system loads into memory is shared by applications.
shared lock	A lock that more than one thread can acquire on the same object. Shared locks allow for greater concurrency with multiple users; if two users have shared locks on a row, a third user cannot change the contents of that row until both users (not just the first) release the lock. Shared-locking strategies are used in all levels of isolation except Informix DIRTY READ and ANSI READ UNCOMMITTED.
shared memory	Memory that is accessible to multiple processes. Shared memory allows multiple processes to access a common data space in memory. Common data does not have to be reread from disk for each process, reducing disk I/O and improving performance. Also used as an Inter-Process Communication (IPC) mechanism to communicate between two processes running on the same computer.
shelf	The location of an optical platter that is neither on an optical drive nor in a jukebox slot.
shuffling	Shuffling refers to the process that occurs when database server moves rows or key values from one fragment to another. Shuffling occurs in a variety of circumstances including when you attach, detach, or drop a fragment.

signal	A means of asynchronous communication between two processes. For example, signals are sent when a user or a program wants to interrupt or suspend the execution of a process. See <i>interrupt</i> .
single-byte character	A character that consists of one byte.
singleton select	A SELECT statement that returns a single row.
SMP	See <i>symmetric multiprocessing system</i> .
source file	A file containing instructions (in ASCII text) that is used as the source for generating compiled programs.
SPL	Acronym for Stored Procedure Language. See <i>stored procedure</i> .
speed up	The ability to add computing hardware to achieve correspondingly faster performance for a DSS query or OLTP operation of a given volume.
SQL	Acronym for Structured Query Language. SQL is a database query language that was developed by IBM and standardized by ANSI. Informix relational database management products are based on an extended implementation of ANSI-standard SQL.
SQL API	An SQL application programming interface that includes a library of callable functions, which are used to develop an application that accesses a relational database. Examples include the embedded-language product INFORMIX-ESQL/C. See <i>host variable</i> .
SQLCA	Acronym for SQL Communications Area. The SQLCA is a data structure that stores information about the most recently executed SQL statement. The result code returned by the database server to the SQLCA is used for error handling by Informix SQL APIs.
sqlda	Acronym for SQL descriptor area. A dynamic SQL management structure that can be used with the DESCRIBE statement to store information about database columns or host variables used in dynamic SQL statements. The structure contains an sqlvar_struct structure for each column; each sqlvar_struct structure provides information such as the name, data type, and length of the column. The sqlda structure is an Informix-specific structure for handling dynamic columns. It is available only within an INFORMIX-ESQL/C program. See also <i>descriptor</i> and <i>system-descriptor area</i> .
sqlhosts	A file that identifies the types of connections the database server supports.

stack operator	Operators that allow programs to manipulate values that are on the stack.
staging-area blobspace	(Not for Dynamic Server with AD and XP Options) The blobspace where a database server temporarily stores BYTE or TEXT data that is being outmigrated to an optical storage subsystem.
statement	A line, or set of lines, of program code that describes a single action (for example, a SELECT statement or an UPDATE statement).
statement block	A section of a program that usually begins and terminates with special symbols such as <i>begin</i> and <i>end</i> . A statement block is the smallest unit of scope of reference for program variables.
statement identifier	An embedded variable name or SQL statement identifier that represents a data structure defined in a PREPARE statement. It is used for dynamic SQL statement management by Informix SQL APIs.
static table	A nonlogging, read-only permanent table.
status variable	A program variable that indicates the status of some aspect of program execution. Status variables often store error numbers or act as flags to indicate that an error has occurred.
stored procedure	A function that is used along with SQL statements in an Informix program. Stored procedures are written using SQL and Stored Procedure Language (SPL) statements. The procedure is stored in the database in executable form.
string	A set of characters (generally alphanumeric) that is manipulated as a single unit. A string might consist of a word (such as 'Smith'), a set of digits representing a number (such as '19543'), or any other collection of characters. Strings generally are surrounded by single quotes. A string is also a character data type, available in INFORMIX-ESQL/C programs, in which the character string is stripped of trailing blanks and is null-terminated.
subordinate table	See <i>outer join</i> .
subquery	A query that is embedded as part of another SQL statement. For example, an INSERT statement can contain a subquery in which a SELECT statement supplies the inserted values in place of a VALUES clause; an UPDATE statement can contain a subquery in which a SELECT statement supplies the updating values; or a SELECT statement can contain a subquery in which a second SELECT statement supplies the qualifying conditions of a WHERE clause for the first SELECT statement. (Parentheses always delimit a subquery, unless you are referring to a CREATE VIEW statement or unions.)

subscript	A subscript is an offset into an array. Subscripts can be used to indicate the start or end position in a CHAR variable.
substring	A portion of a character string.
symmetric multiprocessing system	A symmetric multiprocessing (SMP) system is composed of multiple computers that are connected to a single high-speed communication subsystem. An SMP has fewer computers than an MPP, and cannot be partitioned into nodes.
synonym	A name that is assigned to a table and used in place of the original name for that table. A synonym does not replace the original table name; instead, it acts as an alias for the table.
system call	A call to a function provided by the operating system.
system catalog	A group of database tables that contain information about the database itself, such as the names of tables or columns in the database, the number of rows in a table, the information about indexes and database privileges, and so on. See <i>data dictionary</i> .
system-descriptor area	A dynamic SQL management structure that is used with the ALLOCATE DESCRIPTOR, DEALLOCATE DESCRIPTOR, DESCRIBE, GET DESCRIPTOR, and SET DESCRIPTOR statements to store information about database columns or host variables used in dynamic SQL statements. The structure contains an item descriptor for each column; each item descriptor provides information such as the name, data type, length, scale, and precision of the column. The system-descriptor area is the X/Open standard for handling dynamic columns. See <i>descriptor</i> and <i>sqllda</i> .
system monitoring interface	See <i>SMI</i> .
table	A rectangular array of data in which each row describes a single entity and each column contains the values for each category of description. For example, a table can contain the names and addresses of customers. Each row corresponds to a different customer and the columns correspond to the name and address items. A table is sometimes referred to as a <i>base table</i> to distinguish it from the views, indexes, and other objects defined on the underlying table or associated with it.
table fragment	Zero or more rows that are grouped together and stored in a dbspace that you specify when you create the fragment.

target table	The underlying base table that a violations table and diagnostics table are associated with. You use the START VIOLATIONS TABLE statement to create the association between the target table and the violations and diagnostics tables.
TCP/IP	The specific name of a particular standard transport layer protocol (TCP) and network layer protocol (IP). A popular network protocol used in DOS, UNIX, and other environments.
temp table	A logging temporary table that support indexes, constraints, and rollback.
temporary	An attribute of any file, index, or table that is used only during a single session. Temporary files or resources are typically removed or freed when program execution terminates or an on-line session ends.
TLI	Acronym for Transport Level Interface. It is the interface designed for use by application programs that are independent of a network protocol.
trace	To keep a running list of the values of program variables, arguments, expressions, and so on, in a program or stored procedure.
transaction	A collection of one or more SQL statements that is treated as a single unit of work. If one statement in a transaction fails, the entire transaction can be <i>rolled back</i> (canceled). If the transaction is successful, the work is <i>committed</i> and all changes to the database from the transaction are accepted.
transaction logging	The process of keeping records of transactions. See <i>logical log</i> .
transaction mode	The method by which constraints are checked during transactions. You use the SET statement to specify whether constraints are checked at the end of each data manipulation statement or after the transaction is committed.
trigger	A mechanism that resides in the database. It specifies that when a particular action (insert, delete, or update) occurs on a particular table, the database server should automatically perform one or more additional actions.
tuple	See <i>row</i> .

unbuffered disk I/O	Disk I/O that is controlled directly by the database server instead of the operating system. This direct control helps improve performance and reliability for updates to database data. Unbuffered I/O is supported by character-special files on UNIX and by both unbuffered files and the raw disk interface on Windows NT
unique constraint	Specifies that each entry in a column or set of columns has a unique value.
unique index	An index that prevents duplicate values in the indexed column.
unique key	See <i>primary key</i> .
UNIX real user ID	See <i>Informix user ID</i> .
unload job	The information required to unload data from a Dynamic Server database using the HPL. This information includes format, map, query, device array, project, and special options.
unlock	To free an object (database, table, page, or row) that has been locked. For example, a locked table prevents others from adding, removing, updating, or (in the case of an exclusive lock) viewing rows in that table as long as it is locked. When the user or program unlocks the table, others are permitted access again.
update	The process of changing the contents of one or more columns in one or more existing rows of a table.
update lock	A promotable lock that is acquired during a SELECT...FOR UPDATE. An update lock behaves like a shared lock until the update actually occurs, and it then becomes an exclusive lock. It differs from a shared lock in that only one update lock can be acquired on an object at a time.
user ID	See <i>Informix user ID</i> .
user ID password	See <i>Informix user password</i> .
user name	See <i>Informix user ID</i> .
user password	See <i>Informix user password</i> .

variable	The identifier for a location in memory that stores the value of a program object whose value can change during the execution of the program.
view	A dynamically controlled picture of the contents in a database that allows a programmer to determine what information the user sees and manipulates. A view represents a virtual table based on a specified SELECT statement.
violations table	A special table that holds rows that fail to satisfy constraints and unique index requirements during data manipulation operations on base tables. You use the START VIOLATIONS TABLE statement to create a violations table and associate it with a base table.
virtual column	A derived column of information that is not stored in the database. For example, you can create virtual columns in a SELECT statement by arithmetically manipulating a single column, such as multiplying existing values by a constant, or by combining multiple columns, such as adding the values from two columns.
virtual processors	A virtual processor is a multithreaded process that can serve multiple clients and, where necessary, run multiple threads to work in parallel for a single query.
warning	A state or situation that is detected by the database server or compiler, possibly incorrect syntax or a problem. A warning does not necessarily affect the ability of the code to run.
white space	A series of one or more space characters. The locale file defines what characters are considered to be space characters.
wildcard	A special symbol that represents any sequence of zero or more characters or any single character. In SQL, for example, you can use the asterisk (*), question mark (?), brackets ([]), percent sign (%), and underscore (_) as wildcard characters. (The asterisk, question mark, and brackets are also wildcards in UNIX.)
window	A rectangular area on the screen in which you can take actions without leaving the context of the background program.

WORM	Acronym for Write-Once-Read-Many optical media. When a bit of data is written to a WORM platter, a permanent mark is made on the platter.
X/Open	An independent consortium that produces and develops specifications and standards for open-systems products and technology such as dynamic SQL.
X/Open Portability Guide	A set of specifications which vendors and users can use to build portable software. Any vendor carrying the XPG brand on any particular software product is guaranteeing that the software correctly implements the X/Open Common Applications Environment (CAE) specifications. There are CAE specifications for SQL, XA, ISAM, RDA, and so on.
zoned decimal	A data representation that uses the low-order four bits of each byte to designate a decimal digit (0 through 9) and the high-order four bits to designate the sign of the digit.

Index

A

ALTER TABLE statement, MODIFY
NEXT SIZE clause 1-9

ANSI compliance

- ansi flag 3-25
- DBANSIWARN environment
variable 3-25
- icon Intro-10
- level Intro-17

Archiving

- setting a different ttermcap
file 3-23
- setting DBREMOTECMD to
override default remote
shell 3-40
- setting personal default qualifier
file 3-22

ARC_DEFAULT environment
variable 3-22

ARC_KEYPAD environment
variable 3-23

B

Boolean expression

- with TEXT data type 2-24

Bourne shell, profile file 3-8

Buffer, setting size of fetch

- buffer 3-48

BYTE and TEXT data types

- setting buffer size 3-26

BYTE data type

- description of 2-5
- increasing buffer size 3-26
- inserting data 2-6

location shown in sysblobs
table 1-12

restrictions

- in Boolean expression 2-6
- with GROUP BY 2-5
- with LIKE or MATCHES 2-5
- with ORDER BY 2-5
- selecting a BYTE column 2-6

C

C compiler, setting INFORMIXC
environment variable 3-50

C shell

- .cshrc file 3-8
- .login file 3-8

call_type table in stores7 database,
columns in A-6

CHAR data type

- changing data types 2-27
- collation 2-7
- multibyte values 2-7
- with numeric values 2-7

CHARACTER data type 2-8

Character string

- as DATE values 2-34
- as DATETIME values 2-12, 2-34
- as INTERVAL values 2-18

CHARACTER VARYING data
type, description of 2-8

Check constraint

- described in syschecks table 1-13
- described in syscoldepend
table 1-15

Checking contents of environment
configuration file 3-11

- chkenv utility
 - description of 3-11
 - error message for 3-11
- Client/server
 - INFORMIXSQLHOSTS
 - environment variable 3-56
 - shared memory communication
 - segments 3-55, 3-56
 - specifying default database 3-54
 - specifying stacksize for client
 - session 3-57
 - SQLEXEC environment
 - variable 3-67, 3-68
 - SQLRMDIR environment
 - variable 3-68
- Collation
 - with CHAR data type 2-7
 - with TEXT data type 2-25
 - with VARCHAR data type 2-26
- Colon (:)
 - as delimiter in DATETIME 2-11
 - as delimiter in INTERVAL 2-18
- Color, setting INFORMIXTERM
 - for 3-58
- Column
 - changing data type 2-27
 - constraints, listed in
 - sysconstraints table 1-18
 - defaults, described in sysdefaults
 - table 1-19
 - described in syscolumns
 - table 1-15
 - in stores7 database A-2 to A-7
 - length, shown in syscolumns
 - table 1-16
 - maximum/minimum, shown in
 - syscolumns table 1-18
 - referential constraints in
 - sysreferences table 1-40
- Column-level privilege, described
 - in syscolauth table 1-14
- Command-line conventions
 - elements of Intro-11
 - example diagram Intro-12
 - how to read Intro-12
- Comment icons Intro-8

- Compilation of ESQL/C programs,
 - environment variable to change
 - order of 3-24
- Compiler, environment variable for
 - C 3-50
- Compiling multithreaded ESQL/C
 - applications 3-71
- Compliance
 - icons Intro-10
- Compliance, with industry
 - standards Intro-17
- Configuration file
 - for database servers 3-60
 - for ON-Archive utility 3-22
 - for tctermcap 3-23
- CONNECT statement and
 - INFORMIXSERVER
 - environment variable 3-55
- Connecting to data
 - INFORMIXCONRETRY
 - environment variable 3-50
 - INFORMIXCONTIME
 - environment variable 3-51
- Connection
 - INFORMIXCONRETRY
 - environment variable 3-50
 - INFORMIXCONTIME
 - environment variable 3-51
- Connection authentication, DCE-
 - GSS CSM environment
 - variable 3-53
- Constraint
 - check, described in syscoldepend
 - table 1-15
 - column, described in
 - sysconstraints table 1-18
 - not null, described in
 - syscoldepend table 1-15
 - referential, described in
 - sysreferences table 1-40
- Converting data types 2-27
- CPFIRST environment
 - variable 3-24
- CREATE SCHEMA statement,
 - example 1-4
- CREATE VIEW statement, in
 - sysviews table 1-7

- CURRENT keyword, with
 - DATETIME 2-33
- customer table in stores7
 - database A-2
- cust_calls table in stores7 database,
 - columns in A-6

D

- Data distributions, specifying disk
 - space to use 3-46
- Data type
 - approximate 1-58
 - BYTE 2-5
 - CHAR 2-6
 - CHARACTER 2-8
 - CHARACTER VARYING 2-8
 - conversion 2-27
 - DATE 2-8
 - DATETIME 2-9
 - DEC 2-13
 - DECIMAL 2-13
 - DOUBLE PRECISION 2-15
 - exact numeric 1-58
 - FLOAT 2-15
 - floating-point 2-15
 - INT 2-15
 - INTEGER 2-16
 - INTERVAL 2-16
 - MONEY 2-19
 - NUMERIC 2-21
 - NVARCHAR 2-21
 - REAL 2-21
 - SERIAL 2-21
 - SMALLFLOAT 2-22
 - SMALLINT 2-23
 - summary table 2-3
 - TEXT 2-23
 - VARCHAR 2-25
- Database
 - map of
 - sales_demo A-46
 - stores7 A-8
 - system catalog tables 1-52
 - objects, state, described in
 - sysobjectstate table 1-32
 - sales_demo A-46
 - stores7 description of A-1

- Database server
 - attributes in Information Schema
 - view 1-59
 - setting SQLEXEC 3-67
 - specifying default for
 - connection 3-54
 - SQLRM environment
 - variable 3-68
 - SQLRMDIR environment
 - variable 3-68
- Database, stores7 A-1
- Data-distribution information, in
 - sysdistrib table 1-22
- DATE data type
 - converting to DATETIME 2-29
 - description of 2-8
 - international date formats 2-9
 - range of operations 2-29
 - representing DATE values 2-33
 - two-digit year values and
 - DBCENTURY variable 2-9
 - using with DATETIME and
 - INTERVAL values 2-33
- Date value, setting DBDATE
 - environment variable 3-29
- DATETIME data type
 - adding or subtracting INTERVAL
 - values 2-32
 - character string values 2-12
 - converting to DATE 2-29
 - CURRENT keyword 2-33
 - description of 2-9
 - field qualifiers 2-10
 - formats with DBTIME 3-43
 - international date and time
 - formats 2-12
 - multiplying values 2-31
 - precision and size 2-10
 - range of expressions 2-30
 - range of operations with DATE
 - and INTERVAL 2-29
 - representing DATETIME
 - values 2-33
 - specifying display format 3-43
 - two-digit year values and
 - DBDATE variable 2-12
 - using the DBTIME environment
 - variable 3-43
 - with EXTEND function 2-31, 2-32
- DAY keyword
 - as DATETIME field qualifier 2-10
 - as INTERVAL field qualifier 2-17
- DBANSIWARN environment
 - variable 3-25
- DBBLOBBUF environment
 - variable 3-26
- DBCENTURY environment
 - variable
 - description of 3-26
 - effect on functionality of
 - DBDATE 3-31
- DBDATE environment
 - variable 3-29
- DBDELIMITER environment
 - variable 3-32
- DBEDIT environment variable 3-32
- dbexport utility, specifying field
 - delimiter with
 - DBDELIMITER 3-32, 3-46
- DBFLTMASK environment
 - variable 3-33
- DBLANG environment
 - variable 3-33
- dbload utility, specifying field
 - delimiter with
 - DBDELIMITER 3-32
- DBMONEY environment
 - variable 3-35
- DBONPLOAD environment
 - variable 3-36
- DBPATH environment
 - variable 3-37
- DBPRINT environment
 - variable 3-39
- DBREMOTECMD environment
 - variable 3-40
- Dbserver group, value in
 - INFORMIXSERVER 3-55
- DBSPACETEMP environment
 - variable 3-41
- DBTIME environment
 - variable 3-43
- DBUPSPACE environment
 - variable 3-46
- DCE-GSS communications support
 - module (CSM), environment
 - variable for 3-53
- DEC data type 2-13
- DECIMAL data type
 - changing data types 2-27
 - description of 2-13
 - disk storage 2-14
 - floating-point 2-13
- Decimal digits, display of 3-33
- Decimal point (.)
 - as delimiter in DATETIME 2-11
 - as delimiter in INTERVAL 2-18
- Default configuration file 3-60
- Default locale Intro-4
- Defaults, column, in sysdefaults
 - table 1-19
- DELIMIDENT environment
 - variable 3-46
- Delimited identifier, setting
 - DELIMIDENT environment
 - variable 3-46
- Delimiter
 - for DATETIME values 2-11
 - for INTERVAL values 2-18
- Demonstration database Intro-4
 - map of A-8
 - structure of tables A-2
 - tables in A-2 to A-7
- Diagnostics, for base tables,
 - described in sysviolations
 - table 1-50
- Directives for query optimization,
 - environment variable for 3-48
- Disk space
 - specifying for data
 - distributions 3-46
- Documentation
 - on-line manuals Intro-14
- Documentation conventions
 - command-line Intro-10
 - icon Intro-8
 - sample-code Intro-13
 - typographical Intro-7

- Documentation notes Intro-15
- Documentation notes, program item Intro-16
- Documentation, types of
 - documentation notes Intro-15
 - error message files Intro-14
 - machine notes Intro-15
 - printed manuals Intro-14
 - related reading Intro-16
 - release notes Intro-15

E

- Editor, specifying with
 - DBEDIT 3-32
- ENVIGNORE environment variable
 - description 3-7, 3-47
 - relation to chkenv utility 3-11
- Environment configuration file
 - debugging with chkenv 3-11
- Environment variable
 - ARC_DEFAULT 3-22
 - ARC_KEYPAD 3-23
 - command prompt utilities
 - system environment variables 3-15
 - CPFIRST 3-24
 - DBANSIWARN 3-25
 - DBBLOBBUF 3-26
 - DBCENTURY 3-26
 - DBDATE 3-29
 - DBDELIMITER 3-32
 - DBEDIT 3-32
 - DBFLTMASK 3-33
 - DBLANG 3-33
 - DBMONEY 3-35
 - DBONPLOAD 3-36
 - DBPATH 3-37
 - DBPRINT 3-39
 - DBREMOTECMD 3-40
 - DBSPACETEMP 3-41
 - DBTIME 3-43
 - DBUPSPACE 3-46
 - definition of 3-5
 - DELIMIDENT 3-46
 - ENVIGNORE 3-47
 - FET_BUF_SIZE 3-48
- for command prompt utilities for Windows
 - set in autoexec.bat 3-15
 - set in System applet 3-15
 - set on command-line 3-15
 - user environment variables 3-15
 - variable scope 3-15
- how to set in Bourne shell 3-8
- how to set in C shell 3-8
- how to set in Korn shell 3-8
- IFX_DIRECTIVES 3-48
- INFORMIXC 3-50
- INFORMIXCONRETRY 3-50
- INFORMIXCONTIME 3-51
- INFORMIXDIR 3-52
- INFORMIXKEYTAB 3-53
- INFORMIXOPCACHE 3-54
- INFORMIXSERVER 3-54
- INFORMIXSHMBASE 3-55, 3-56
- INFORMIXSQLHOSTS 3-56
- INFORMIXSTACKSIZE 3-57
- INFORMIXTERM 3-58
- INF_ROLE_SEP 3-59
- listed, by topic 3-71
- modifying 3-9
- NODEFDAC 3-59
- ONCONFIG 3-60
- OPTCOMPIND 3-61
- overriding a setting 3-7, 3-47
- PATH 3-62
- PDQPRIORITY 3-63
- PLCONFIG 3-64
- PSORT_DBTEMP 3-64
- PSORT_NPROCS 3-65
- rules of precedence 3-12, 3-17
- setting
 - at the command line 3-6
 - for command prompt utilities 3-14
 - for native Windows NT applications 3-13
 - in a login file 3-6
 - in a shell file 3-8
 - in an environment-configuration file 3-6
 - in autoexec.bat 3-15
 - on the command-line 3-15
 - using the Registry Editor 3-13

- using the System applet 3-15
- SQLEXEC 3-67
- SQLRM 3-68
- SQLRMDIR 3-68
- TERM 3-69
- TERMCAP 3-69
- TERMINFO 3-70
- THREADLIB 3-71
- types of 3-5
- variable scope 3-15
- view current setting 3-10
- where to set 3-8
- Environment variables
 - Manipulating in Windows environments 3-13
 - setting in Windows environments 3-12
- en_us.8859-1 locale Intro-4
- Error message files Intro-14
- ESQL/C program, compilation order 3-24
- ESQL/C, compiling multithreaded applications 3-71
- Executable programs, where to search 3-62
- EXTEND function, with DATE, DATETIME and INTERVAL 2-31, 2-32
- Extension checking, specifying with DBANSIWARN 3-25
- Extension, to SQL
 - symbol for Intro-10
- Extent, changing size of system table 1-9

F

- Feature icons Intro-9
- Features, produc Intro-5
- FET_BUF_SIZE environment variable 3-48
- Field delimiter files,
 - DBDELIMITER 3-32
- Field qualifier for DATETIME 2-10
- File
 - shell 3-8
 - temporary for database server 3-41

- temporary, sorting 3-64
- termcap, terminfo 3-58, 3-69, 3-70
- with environment configuration information 3-11
- finderr utility Intro-15
- FLOAT data type
 - changing data types 2-27
 - description of 2-15
- Format
 - specifying for DATE value with DBDATE 3-30
 - specifying for DATETIME value with DBTIME 3-43
 - specifying for MONEY value with DBMONEY 3-35
- FRACTION keyword
 - as DATETIME field qualifier 2-10
 - as INTERVAL field qualifier 2-17
- Fragmentation
 - information in sysfragments table 1-27
 - setting priority levels for PDQ 3-62

G

- Global Language Support (GLS) Intro-4

H

- High-Performance Loader, environment variable for 3-36, 3-64
- HOURL keyword
 - as DATETIME field qualifier 2-10
 - as INTERVAL field qualifier 2-17
- Hyphen (-)
 - as delimiter in DATETIME 2-11
 - as delimiter in INTERVAL 2-18

I

- Icons
 - comment Intro-8
 - compliance Intro-10
 - feature Intro-9

- platform Intro-9
- product Intro-9
- IFX_DIRECTIVES environment variable 3-48
- Index
 - descriptions in sysindexes table 1-29
 - threads for 3-66
- Industry standards, compliance with Intro-17
- Information Schema views
 - accessing 1-56
 - description of 1-55
 - generating 1-55
 - server_info view 1-59
- Informix extension checking, specifying with DBANSIWARN 3-25
- INFORMIXC environment variable 3-50
- INFORMIXCONRETRY environment variable 3-50
- INFORMIXCONTIME environment variable 3-51
- INFORMIXDIR environment variable 3-52
- INFORMIXDIR/bin directory Intro-5
- INFORMIXKEYTAB environment variable 3-53
- INFORMIXOPCACHE environment variable 3-54
- INFORMIXSERVER environment variable 3-54
- INFORMIXSHMBASE environment variable 3-55, 3-56
- INFORMIXSTACKSIZE environment variable 3-57
- INFORMIXTERM environment variable 3-58
- INF_ROLE_SEP environment variable 3-59
- Inserting, values into SERIAL columns 2-22
- Installation directory, specifying with INFORMIXDIR 3-53
- Installation files, INFORMIXDIR environment variable 3-52

- INTEGER data type
 - changing data types 2-27
 - description of 2-16
- Intensity attributes, setting INFORMIXTERM for 3-58
- INTERVAL data type
 - adding or subtracting from 2-34
 - adding or subtracting from DATETIME values 2-32
 - description of 2-16
 - field delimiters 2-18
 - multiplying or dividing values 2-35
 - range of expressions 2-30
 - range of operations with DATE and DATETIME 2-29
 - with EXTEND function 2-31, 2-32
- ISO 8859-1 code set Intro-4
- items table in stores7 database, columns in A-4

K

- keytab file, environment variable to specify path 3-53
- Korn shell, .profile file 3-8

L

- Language environment
 - DBLANG 3-33
 - setting with DBLANG 3-33
- LOAD statement
 - specifying field delimiter with DBDELIMITER 3-32
- Locale Intro-4

M

- Machine notes Intro-15
- Memory cache, for Optical StageBlob area 3-54
- Message file
 - error messages Intro-14
- Message files, specifying subdirectory with DBLANG 3-34

MINUTE keyword
 as DATETIME field qualifier 2-10
 as INTERVAL field qualifier 2-17

MONEY data type
 changing data types 2-27
 description of 2-19
 display format specified with
 DBMONEY 3-35
 international money formats 2-20

MONTH keyword
 as DATETIME field qualifier 2-10
 as INTERVAL field qualifier 2-17

Multibyte characters, with CHAR
 data type 2-7

N

Network environment variable
 DBPATH 3-37
 SQLRM 3-68
 SQLRMDIR 3-68

NODEFDAC environment variable,
 description of 3-59

Nonprintable characters, with
 CHAR data type 2-7

Not null constraint, described in
 syscoldepend table 1-15

NULL value
 testing in BYTE expression 2-6
 testing with TEXT data type 2-24

Null value, in columns, status in
 syscolumns table 1-16

NUMERIC data type 2-21

O

Object mode of database objects,
 described in sysobjstate
 table 1-32

ONCONFIG environment
 variable 3-60

On-line manuals Intro-14

OPTCOMPIND environment
 variable, values 3-61

Optical cluster, described in
 sysopclstr table 1-33

Optical StageBlob area, memory
 cache for 3-54

orders table in stores7 database,
 columns in A-3

P

Parallel distributed queries, setting
 with PDQPRIORITY
 environment variable 3-62

Parallel sorting, using
 PSORT_NPROCS for 3-64

PATH environment variable 3-62

Pathname
 for C compiler 3-50
 for database server 3-37
 for executable programs 3-62
 for installation 3-52
 for message files 3-33
 for parallel sorting 3-64
 for relay module 3-67
 for remote shell 3-40
 specifying with DBPATH 3-37
 specifying with PATH 3-62

PDQ
 OPTCOMPIND environment
 variable 3-61
 PDQPRIORITY environment
 variable 3-62

Platform icons Intro-9

PLCONFIG environment
 variable 3-64

Precedence, rules for command-line
 utility environment
 variables 3-17

Precedence, rules for environment
 variables for Informix native
 Windows applications 3-18

Precedence, rules for environment
 variables in UNIX 3-12

Prepared statement
 version number in systables 1-46

Printed manuals Intro-14

Printing with DBPRINT 3-40

Privilege
 on a table fragment, described in
 sysfragauth table 1-25

on a table, described in
 systabauth 1-43

on database, described in the
 sysusers table 1-49

preventing to PUBLIC 3-59

related to procedures, described
 in sysprocauth table 1-36

user, described in sysusers
 table 1-49

Procedure privileges, described in
 sysprocauth table 1-36

Product icons Intro-9

Program group
 Documentation notes Intro-16
 Release notes Intro-16

Protected stored procedures 1-38

PSORT_DBTEMP environment
 variable 3-64

PSORT_NPROCS environment
 variable 3-65

Q

Qualifier field, for DATETIME 2-10

Query optimization information in
 sysprocplan table 1-39

Query optimizer, environment
 variable for directives 3-48

R

REAL data type 2-21

regedt32.exe 3-13

Related reading Intro-16

Relay module
 SQLRM environment
 variable 3-68
 SQLRMDIR environment
 variable 3-68

Release notes Intro-15

Release notes, program
 item Intro-16

Remote shell 3-40

Role separation, environment
 variable for 3-59

Role, granted to user, described in
 sysroleauth table 1-41

Routine DATETIME
 formatting 3-43
Runtime program, setting
 DBANSIWARN 3-25

S

sales_demo database A-46
Sample-code conventions Intro-13
SECOND keyword
 as DATETIME field qualifier 2-10
 as INTERVAL field qualifier 2-17
Sequential integers, with SERIAL
 data type 2-21
SERIAL data type
 description of 2-21
 inserting values 2-22
 resetting values 2-22
setenv.cmd 3-17
Setting environment variables 3-7,
 3-13
Shared memory, setting with
 INFORMIXSHMBASE 3-55
Shell
 remote 3-40
 search path 3-62
 setting environment variables in a
 file 3-8
 specifying with
 DBREMOTECMD 3-40
Single-precision floating-point
 number, storage of 2-15
SMALLFLOAT data type
 changing data types 2-27
 description of 2-22
SMALLINT data type
 changing data types 2-27
 description of 2-23
Software dependencies Intro-4
Sorting
 DBSPACETEMP environment
 variable 3-41
 PSORT_DBTEMP environment
 variable 3-64
 setting PSORT_NPROCS
 environment variable 3-65
 threads for 3-65

Space ()
 as delimiter in DATETIME 2-11
 as delimiter in INTERVAL 2-18
SQL Communications Area
 (SQLCA)
 effect of setting
 DBANSIWARN 3-25
SQLEXEC environment
 variable 3-67
SQLRM environment variable 3-68
SQLRMDIR environment
 variable 3-68
Stacksize, setting
 INFORMIXSTACKSIZE 3-57
state table in stores7 database,
 columns in A-7
Statement, SQL
 ANSI compliance and
 DBANSIWARN 3-25
 CONNECT and
 INFORMIXSERVER 3-55
 editing and DBEDIT 3-32, 3-33,
 3-36
 LOAD and DBDELIMITER 3-32,
 3-46
 printing and DBPRINT 3-39
 UNLOAD and
 DBDELIMITER 3-32, 3-46
 UPDATE STATISTICS and
 DBUPSPACE 3-46
stock table in stores7 database,
 columns in A-5
Stored procedure
 characteristics in sysprocedures
 table 1-38
 protected 1-38
stores7 database Intro-4
 call_type table columns A-6
 catalog table columns A-5
 customer table columns A-2
 cust_calls table columns A-6
 data values A-16
 description of A-1
 items table columns A-4
 manufact table columns A-7
 map of A-8
 orders table columns A-3
 primary-foreign key
 relationships A-9 to A-16

stock table columns A-5
structure of tables A-2
Synonym
 for each table view, described in
 sys synonyms table 1-42
 for each table, described in
 sys syntable 1-42
syscolauth catalog table,
 example 1-7
syscolauth system catalog table 1-7
syscolumns system catalog table,
 example 1-6
sysextcols system catalog table 1-23
sysextdfiles system catalog
 table 1-24
sysexternal system catalog
 table 1-24
sysfragauth system catalog table,
 example 1-25
sysindexes system catalog table,
 example 1-8
sysmaster database,
 initialization 3-55
sysobjstate system catalog table,
 example 1-32
sysroleauth system catalog table,
 example 1-41
systabauth system catalog table 1-7
systables system catalog table,
 example 1-5
System catalog
 accessing 1-9
 altering contents 1-9
 description of 1-3
 how used by database server 1-4
 list of tables 1-10
 map of tables 1-52
sysblobs 1-12
syschecks 1-13
syscolauth 1-14
syscoldepend 1-15
syscolumns 1-15
sysconstraints 1-18
sysdefaults 1-19
sysdepend 1-21
sysdistrib 1-22
sysextdfiles table 1-24
sysexternal table 1-24
sysfragauth 1-25

- sysfragments 1-27
- sysindexes 1-29
- sysobjstate 1-32
- sysopclstr 1-33
- sysprocauth 1-36
- sysprocbody 1-36
- sysprocedures 1-38
- sysprocplan 1-39
- sysreferences 1-40
- sysroleauth 1-41
- sysssynonyms 1-42
- sysssyntable 1-42
- systabauth 1-43
- systables 1-44
- sysstrigbody 1-47
- sysstriggers 1-48
- sysusers 1-49
- sysviews 1-50
- sysviolations 1-50
- updating statistics 1-9
- updating system catalog tables 1-9
- using 1-4
- System catalog tables
 - sysextools table 1-23
- sysviews catalog table, example 1-6
- sysviolations systems catalog table, example 1-50

T

- tabid, description of 1-6
- Table
 - changing the data type of a column 2-27
 - dependencies, in sysdepend table 1-21
 - description in systables catalog table 1-44
 - structure in stores7 database A-2
 - synonyms in sysssyntable table 1-42
 - system catalog tables 1-12 to 1-50
- Table-level privilege, shown in tabauth table 1-7
- tabtype 1-45

- Tape management
 - setting ARC_DEFAULT 3-22
 - setting ARC_KEYPAD 3-23
 - setting DBREMOTECMD 3-40
- Temporary files
 - in database server, setting DBSPACETEMP 3-41
 - setting PSORT_DBTEMP 3-64
- Temporary tables
 - specifying dbspace with DBSPACETEMP 3-41
- TERM environment variable 3-69
- TERMCAP environment variable 3-69
- termcap file, and TERMCAP environment variable 3-69
- Terminal handling
 - and TERM environment variable 3-69
 - and TERMCAP environment variable 3-69
 - and TERMINFO environment variable 3-70
 - setting INFORMIXTERM 3-58
- terminfo directory 3-70
- TERMINFO environment variable 3-70
- TEXT data type
 - collation 2-25
 - description of 2-23
 - increasing buffer size 3-26
 - inserting values 2-24
 - location shown in sysblobs table 1-12
 - restrictions
 - with aggregate functions 2-24
 - with GROUP BY 2-24
 - with IN clause 2-24
 - with LIKE or MATCHES 2-24
 - with ORDER BY 2-24
 - selecting a column 2-24
 - use in Boolean expression 2-24
 - with control characters 2-23
- Text editor, specifying with DBEDIT 3-32
- THREADLIB environment variable 3-71
- Time value, setting DBTIME environment variable 3-43

- Trigger
 - information in sysstriggers table 1-48
 - text in sysstrigbody table 1-47

U

- Unique numeric code, with SERIAL data type 2-22
- UNIX
 - BSD, default print capability 3-40
 - environment variables 3-5
 - PATH environment variable 3-62
 - specifying directories for intermediate writes 3-65
- System V
 - default print capability 3-40
 - terminfo library support 3-58
 - TERM environment variable 3-69
 - TERMCAP environment variable 3-69
 - TERMINFO environment variable 3-70
- UNLOAD statement, specifying field delimiter with DBDELIMITER 3-32
- UPDATE STATISTICS statement and DBUPSPACE environment variable 3-46
- effect on sysdistrib table 1-22
- update system catalog 1-9
- User privileges, described in sysusers table 1-49
- Utility program, chkenv 3-11

V

- VARCHAR data type, collation 2-26
- View
 - dependencies, in sysdepend table 1-21
 - described in sysviews table 1-50
 - synonyms in sysssynonyms table 1-42
 - system catalog table 1-50

Violations, for base tables,
described in sysviolations
table 1-50

W

Windows environments
manipulating environment
variables 3-13
setting environment
variables 3-12

X

X/Open
and server_info view 1-59
Information Schema views 1-55
X/Open compliance
level Intro-17
X/Open-compliant databases 1-59

Y

YEAR keyword
as DATETIME field qualifier 2-10
as INTERVAL field qualifier 2-17
Year values, two and four
digit 3-26

Symbols

(), space, as delimiter
in DATETIME 2-11
in INTERVAL 2-18
-, hyphen, as delimiter
in DATETIME 2-11
in INTERVAL 2-18
., decimal point, as delimiter
in DATETIME 2-11
in INTERVAL 2-18
/etc/termcap 3-70
:, colon, as delimiter
in DATETIME 2-11
in INTERVAL 2-18

